Proračun tranzijentne struje kratkog spoja sinkronog stroja s permanentnim magnetima korištenjem mapa ulančenih tokova

Gaži, Kristijan

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 148

PRORAČUN TRANZIJENTNE STRUJE KRATKOG SPOJA SINKRONOG STROJA S PERMANENTNIM MAGNETIMA KORIŠTENJEM MAPA ULANČENIH TOKOVA

Kristijan Gaži

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 148

PRORAČUN TRANZIJENTNE STRUJE KRATKOG SPOJA SINKRONOG STROJA S PERMANENTNIM MAGNETIMA KORIŠTENJEM MAPA ULANČENIH TOKOVA

Kristijan Gaži

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 148

Pristupnik:	**Kristijan Gaži (0036521699)**
Studij:	Elektrotehnika i informacijska tehnologija
Profil:	Elektrostrojarstvo i automatizacija
Mentor:	prof. dr. sc. Stjepan Stipetić

Zadatak: **Proračun tranzijentne struje kratkog spoja sinkronog stroja s permanentnim magnetima korištenjem mapa ulančenih tokova**

Opis zadatka:

Proučite suvremenu literaturu o zadanom problemu. U postojećem MATLAB programskom kodu za određivanje magnetostatičkih mapa ulančenih tokova u d-q osima ovisnih o strujama u d-q osima omogućite automatsko simuliranje različitih topologija sinkronih strojeva s permanentnim magnetima (SPM, IPM, IPM-spoke) uz korištenje simetrije, periodičnosti i paralelizacije. Detaljno dokumentirajte izvorni programski kod. Na temelju dobivenih mapa i njihovih inverznih formi provedite proračun najgoreg slučaja struje kratkog spoja za više različitih topologija iterativnom metodom sekante, diskretiziranim dinamičkim modelom u Matlab skripti, diskretiziranim dinamičkim modelom u Matlab Simulinku te usporedite rezultate s programskim paketom SyR-e, Simcenter Magnet i ANSYS Motor-CAD. Odredite preporuke za brzu i točnu provjeru demagnetizacije sinkronog stroja s permanentnim magnetima.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se mentoru prof. dr. sc. Stjepanu Stipetiću na ustrajnoj i vrijednoj pomoći u izradi ovog rada.

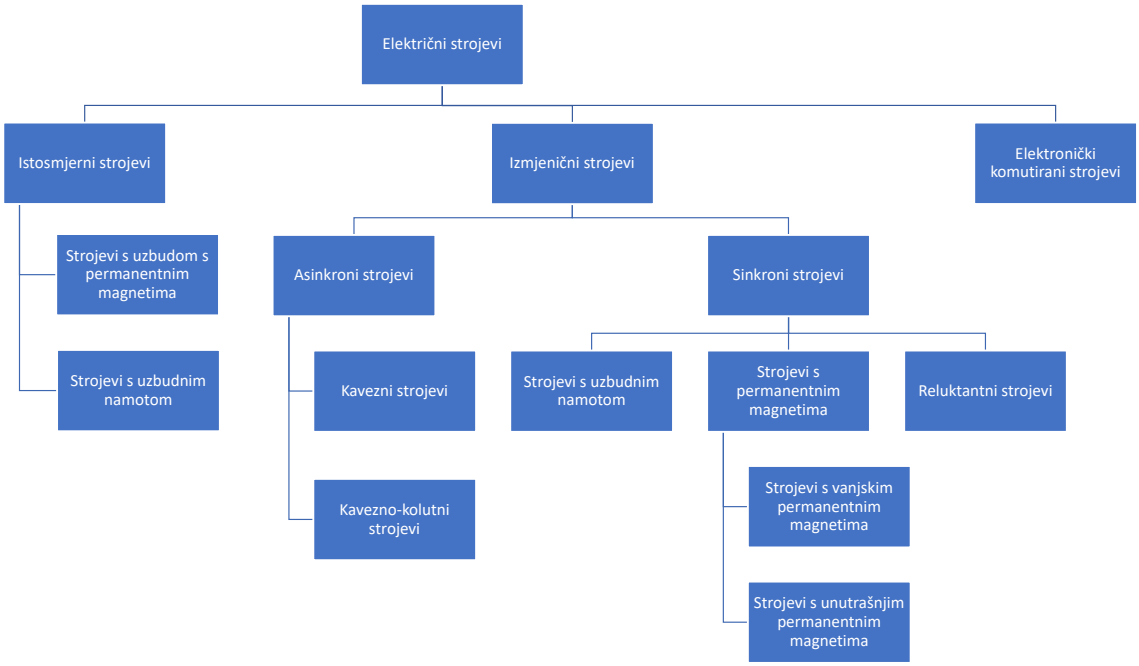
Veliko hvala mojim kolegama s kojima sam podijelio iskustvo studiranja i stvorio mnogo lijepih uspomena. Prijateljima i prijateljicama iz ansambla upućujem zahvale za svu dobru zabavu i odlično provedeno slobodno vrijeme.

Mojoj obitelji, koja me uvijek pratila i hrabrila tijekom studiranja, posebna zahvala. Bez vas ne bih uspio. Na kraju bih se zahvalio dragom Bogu, čijom milošću mi je sve omogućeno.

Sadržaj

1. Uvod

Električni strojevi mogu se podijeliti na izmjenične, istosmjerne i elektronički komutirane strojeve, a izmjenični strojevi dalje se dijele na sinkrone i asinkrone strojeve. Sinkroni strojevi s uzbudom, strojevi s permanentnim magnetima i reluktantni strojevi zajedno čine grupu sinkronih strojeva, kako prikazuje dijagram podjele na slici 1.1.

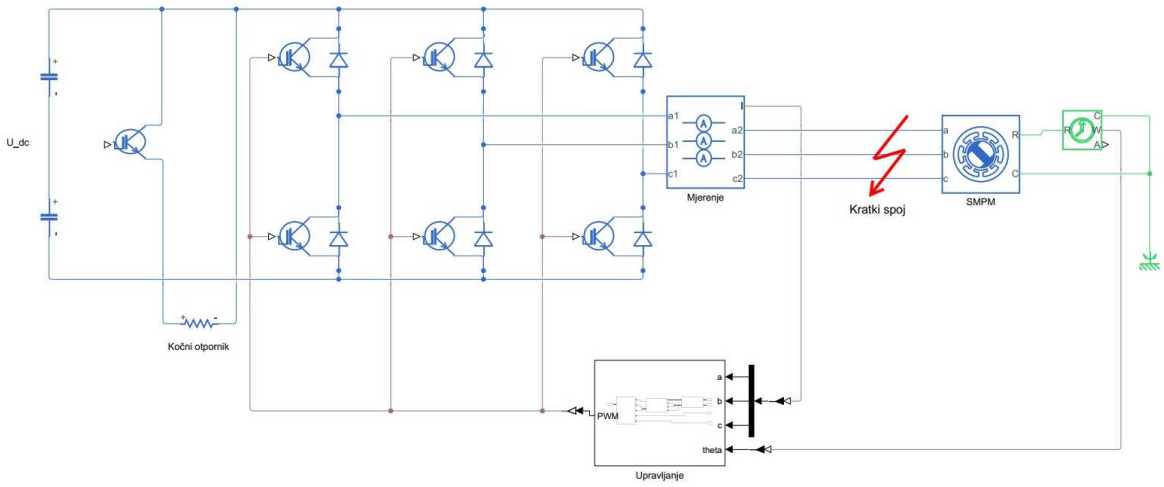


Slika 1.1. Podjela električnih strojeva

Svim sinkronim strojevima zajednički je sinusoidalni valni oblik struje i napona te sinkrona brzina vrtnje rotora, što znači da se rotor i rezultantno magnetsko polje u stroju vrte jednakom brzinom tj. *sinkrono*. Sinkroni strojevi s permanentnim magnetima imaju veću gustoću snage i momenta i višu korisnost u usporedbi s drugim strojevima [1]. Te prednosti donose permanentni magneti koji su najskuplji dio stroja i zato je cijena sinkronih strojeva s permanentnim magnetima relativno veća od cijena ostalih strojeva istog ranga snage. Ovisno o položaju magneta na rotoru mijenjaju se vanjske karakteristike

stroja. Sinkroni strojevi s površinskim permanentnim magnetima imaju ograničen raspon brzina i tipično bolju korisnost, dok sinkroni strojevi s unutrašnjim permanentnim magnetima imaju mogućnost rada u širem rasponu brzina vrtnje i bolje performanse u području slabljenja polja.[2]

Sinkroni strojevi kod kojih se zahtijeva vrtnja konstantnom brzinom mogu se napajati direktno iz mreže. Pogoni promjenjive brzine vrtnje zahtijevaju upravljanje i sastoje se od pretvarača napona i frekvencije, električnog stroja, senzora i upravljačkih jedinica i svaka od tih jedinica izložena je termičkim, mehaničkim i elektromagnetskim naprezanjima. Kvar bilo koje jedinice može uzrokovati kvar cijelog pogona. Osim mehaničkoga, kvar na električnom motoru može biti izazvan kratkim spojem [3]. Slika 1.2. prikazuje shemu pogona sa svim svojim komponentama te je naznačeno mjesto kratkog spoja. Osim što je kratki spoj štetno stanje za namot i izolaciju, može negativno utjecati na magnete. Ako magneti nisu projektirani da izdrže udarne struje kratkog spoja, dolazi do tzv. demagnetizacije. Time permanentni magneti ili djelomično ili potpuno gube svoja magnetska svojstva što negativno utječe na performanse stroja.[1]



Slika 1.2. Elektromotorni pogon sa sinkronim motorom s permanentnim magnetima (SMPM)

U slučaju kratkog spoja na stezaljkama stroja napon naglo postaje jednak nuli i povećavaju se struje, koje su ograničene otporom i induktivitetom namota. Druga vrsta kratkog spoja je međuzavojni kratki spoj. Među vodičima u stroju se može dogoditi kratki spoj zbog oslabljene izolacije, čime se mijenja broj zavoja. U tom slučaju se pojavljuje poremećaj u simetričnosti napona, koji ovisi o intenzitetu kvara. Kratki spoj na stezaljkama uzrokovat će više iznose struja od kratkog spoja među zavojima. U slučaju visokih

iznosa struje, pojavljuju se visoki iznosi tokova koji se suprotstavljaju toku magneta. Što su tokovi u kratkom spoju veći, to će potencijalno biti veći stupanj oštećenja magneta. Za potpunu demagnetizaciju potrebno je značajno više protjecanja suprotstavljenog magnetima, nego za slučaj djelomične demagnetizacije.[4] U najgorem slučaju kratkog spoja koji rezultira potpunom demagnetizacijom, stroj prestaje s radom i potreban je remont, tj. ugradnja ili magnetizacija novih magneta odnosno ugradnja novog rotora. No, mirovanje pogona donosi financijske gubitke, a zamjena magneta dodatne troškove. Stoga je važno provesti simulacijsku analizu kvarnih stanja stroja.[5] U ovom radu kratki spoj je analiziran analitički i simulacijski, a struja potrebna za demagnetizaciju određena je metodom sekante [6] u kojoj se nizom FEM analiza u Simcenter MAGNET-u [7] određuje struja na kojoj se postiže tok jednakog iznosa toku magneta, ali u suprotnom smjeru.

Pregledna tablica (*eng. LUT Look-Up Table*) niz je podataka koji preslikava, tj. mapira ulazne vrijednosti u izlazne vrijednosti, čijim se preslikavanjem aproksimira matematička funkcija. S obzirom na domenu ulaznih vrijednosti, operacija pretraživanja dohvaća odgovarajuće izlazne vrijednosti iz pregledne tablice. Na razne se načine mogu vrijednosti u tablici interpolirati pri kreiranju tablice te ekstrapolirati pri dohvaćanju podataka iz tablice. Stoga pregledne tablice imaju matričnu formu. U kontekstu električnih strojeva često se za preglednu tablicu koristi naziv mapa te će u ovom radu biti korišten taj naziv. Mape ulančenih tokova matrični su prikaz ovisnosti toka o struji u $dq\theta$ sustavu. Dakle, 2D matrica sastoji se od vrijednosti toka, gdje struja I_d opisuje retke, a struja I_q stupce. Dodatno, matrici se može dodati treća dimenzija - pozicija rotora. 3D mape ulančenih tokova dobivaju se provedbom niza simulacija prema uređenoj trojki (i_d^m, i_q^n, θ^j). Za svaku kombinaciju struja i_d^m i i_q^n i kuta rotora θ^j provodi se FEM analiza (*eng. Finite Element Method*) u toj točki. Tako detaljne mape koriste se za analizu korisnosti, upravljanje koje zahtijeva poznavanje dinamike stroja te analize valovitosti momenta. Tijekom izrade ovog rada početno su korištene 3D mape, no 2D mape pokazale su se jednako dobre, ali sa značajno kraćim vremenom izrade. U potpoglavlju 2.2.3. bit će detaljnije objašnjena razlika u korištenju 2D i 3D mapa. Pri zadavanju raspona struja i položaja rotora potrebno je uzeti u obzir kasniju uporabu mape. Za proračun tranzijenta kratkog spoja potrebno je za granice uzeti 5-9 puta veću struju od nazivne kako bi se obuhvatio tranzijent kratkog spoja. Što se tiče pozicija rotora za 3D mape, dovoljno je uzeti jedan prapol stroja jer se na ostatku stroja pojavljuje simetrija. [8] Osim graničnih vrijednosti

mape, važno je od koliko elemenata se sastoji. Težnja je imati što točniju mapu, no pritom se suprotstavljaju simulacijsko vrijeme potrebno za izradu mape i točnost mape te je potrebno odrediti prioritete.

Rezultat FEM simulacije su ulančeni tokovi u abc sustavu, ovisni o kutu rotora, koji se inverznom Parkovom transformacijom preračunava u $dq\theta$ sustav te pohranjuju u tzv. *direktne mape* $\mathbf{\Psi_d}(i_d^m, i_q^n)$ i $\mathbf{\Psi_q}(i_d^m, i_q^n)$. Osim toka, u svakoj se točki sprema iznos momenta u mapu $\mathbf{T}(i_d^m, i_q^n)$.[8] Za izradu mapa u Matlabu postoji funkcija *griddedInterpolant* kojom se podaci povezuju u mapu tako da ih kasnije Matlab može obrađivati u Simulinku u bloku *2D LUT* (*eng. Look-Up Table*). Za proračun dinamičkih modela ili simulaciju upravljanja strojem, potrebne su tzv. *inverzne mape* stroja: $\mathbf{I_d}(\psi_d^m, \psi_q^n)$ i $\mathbf{I_q}(\psi_d^m, \psi_q^n)$. Poznavanje direktnih i inverznih mapa stroja omogućuje izračun mapa korisnosti [9], tranzijente struja u kratkim spojevima [10], analizu napona praznog hoda, analizu valovitosti momenta u različitim radnim točkama i slično.

Cilj korištenja mapa je skraćivanje vremena izvođenja proračuna. Mana provedbe FEM analize je dugo vrijeme izvođenja, koje usporava proces analize stroja. Korištenjem mapa, pojedini proračun traje nekoliko sekundi.[6] Iako je izrada mapa dugotrajna, kasnije je to vrijeme nadoknađeno brzim proračunima i brzim dolaskom do rješenja. FEM analiza je specifična i za točno jednu svrhu, što je mana u odnosu na mape, koje imaju više mogućnosti upotrebe. U ovom radu bit će prezentirana Eulerova metoda za brzo određivanje kratkog spoja [6] u kojoj se prema naponskim jednadžbama u dq sustavu računa tok te se iz inverznih mapa struja po ulančenom toku određuju dq komponente struja tranzijenta kratkog spoja.

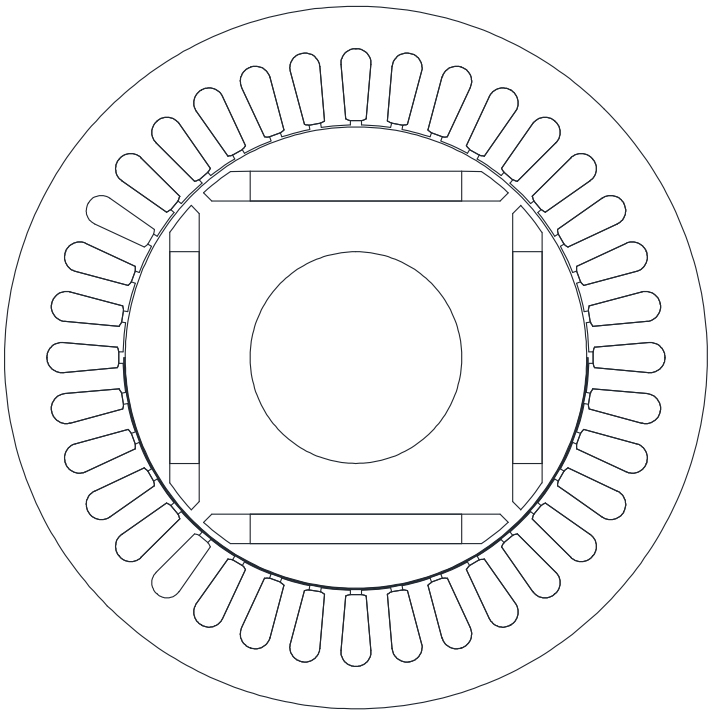
U drugom poglavlju dokumentiran je programski kod u Matlabu. Opisane su skripte za zadavanje parametara stroja, izradu direktnih mapa ulančenih tokova, izračun inverznih strujnih mapa, vizualizaciju mapa, provedbu simulacije kratkog spoja, analitičko određivanje tranzijenta kratkog spoja i izračun struje demagnetizacije metodom sekante. Načinima određivanja tranzijenta kratkog spoja posvećeno je treće poglavlje, gdje su pojedinačno opisani i uspoređeni simulacijski i analitički dobiveni rezultati. Prema podacima o tranzijentu struja kratkog spoja, provodi se metoda sekante i provjerava demagnetizacija te su dobiveni rezultati prikazani u četvrtom poglavlju. Peto poglavlje prezentira rezultate analize na drugačijoj topologiji stroja, kao pokazatelj općenitosti i funkcional-

nosti napisanog koda. U zadnjem poglavlju donose se zaključci o provedenim načinima izračuna tranzijenta i komentirane su implementirane metode iz [6]. Glavna ideja diplomskog rada zasnovana je na članku [6] sa Sveučilišta u Torinu, provjerene su i implementirane predložene metode te će biti prezentirani dobiveni rezultati.

Stroj na kojem je kod razvijen i na kojem su simulirane sve funkcije je Allwin PM-AW112M5-4-E, nazivnih podataka napisanih u tablici 1.1. Motor je četveropolni sinkroni stroj s unutrašnjim permanentnim magnetima i u ostatku rada se kraće naziva IPM motor. Najvažniji podatak u analizi kratkog spoja je nazivna struja koja iznosi 11,6 A_{RMS}, odnosno vršne vrijednosti 16,4 A_{pk}. Poprečni presjek stroja prikazan je na slici 1.3.

Allwin	3PH AC Frequency Driven PM Motor	
Type: PM-AW112M5-4-E	IP 45	SN: 0/1081WAW029-2016
380 V	5,5 kW 11,6 A	1500 min^{-1}
Ef. 92,1% 4P	50 Hz Y conn.	In.cl. F
BEMF 220 V/krpm	Ind. 69 mH	duty S1
30 kg	No: 0002	Date: 17/06

Tablica 1.1. Natpisna pločica IPM motora



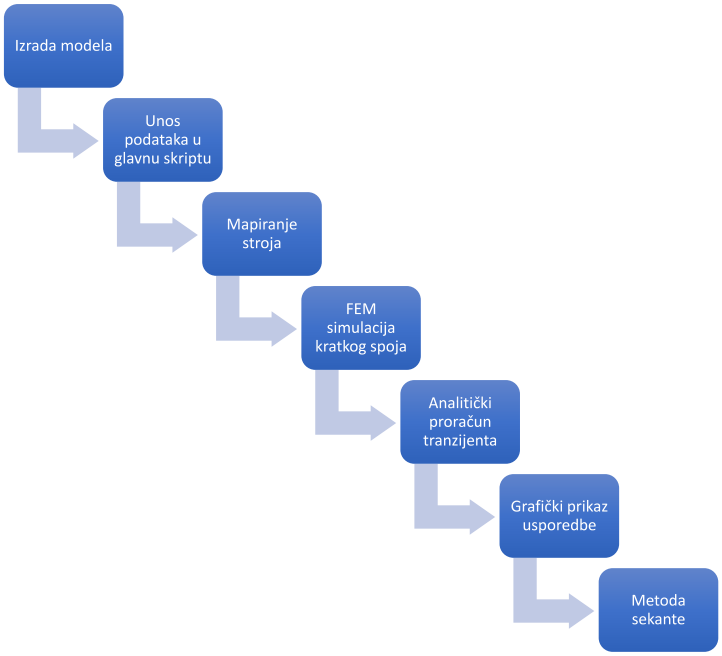
Slika 1.3. Poprečni presjek IPM motora

2. Programski kod, tijek proračuna i upute za izradu modela

Programski alat je koncipiran tako da korisnik unese potrebne parametre u glavnu skriptu stroja. Uz to je potrebno priložiti modele stroja, napravljene prema uputi kako bi Matlab mogao ispravno prepoznati model. Ovisno o tome što korisniku treba, mogu se neovisno pokretati dijelovi glavne skripte, no korisnik ne treba poznavati sve funkcije koje glavna skripta poziva i koje se izvršavaju u pozadini. U ovom poglavlju na početku je objašnjena glavna skripta te su redom objašnjeni dijelovi koda kako bi se dao detaljan uvid u programiranje svake funkcionalnosti. Glavna ideja je spremati rezultate svakog dijela proračuna u zasebne datoteke. Svi podaci organizirani su po strukturama, koje su spremljene u *.mat* datoteke. Postoji datoteka s podacima o stroju (npr. nazivni podaci napona i struje, broj polova, frekvencija, otpor...) i podacima o modelu stroja (npr. broj perioda i vremenski korak za tranzijent kratkog spoja) i ona je spremljena na početku proračuna. Kasnije se stvaraju datoteke za: direktnu mapu, inverznu mapu, tranzijent kratkog spoja iz FEM analize i Eulerove metode te rezultati metode sekante.

2.1. Tijek proračuna

Da bi se bolje razumio slijed funkcija u glavnoj skripti, ovo je potpoglavlje posvećeno tijeku simulacija 2.1. Korisnik najprije treba izraditi modele prema uputama u poglavlju 2.3. Na početku glavne skripte predviđen je prostor za unos traženih podataka o stroju nakon čega skripta dalje automatski provodi cijeli proračun. Redom se rade direktna i inverzna mapa stroja, provodi se simulacija kratkog spoja u Simcenter MAGNET-u i analitički proračun tranzijenta kratkog spoja čiji se rezultati uspoređuju grafički. Na kraju se metodom sekante, nepovezanom s mapama, provjerava može li se rotor demagnetizirati zbog kratkog spoja.



Slika 2.1. Tijek simulacija

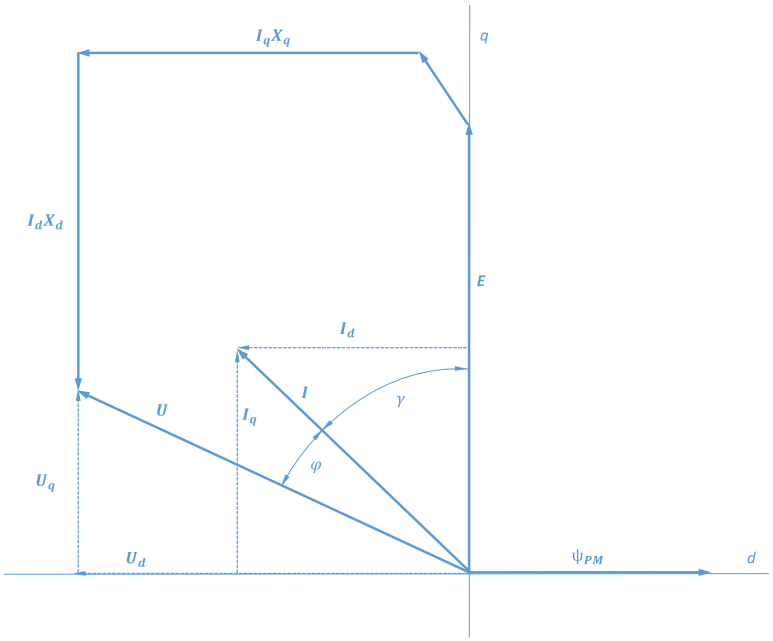
2.2. Programski kod

Programski kod napisan je na engleskom jeziku i sve će varijable i funkcije imati nazive na engleskom jeziku. Grafovi prikazani u ovom radu imaju hrvatske naslove i oznake, no originalni kod je u potpunosti na engleskom.

2.2.1. Glavna skripta

Desetak sekcija čine glavnu skriptu u Matlabu, omogućujući da se svaka funkcija izvrši neovisno. Neke funkcije, primjerice funkcija za izradu direktne mape, uzimaju puno vremena pa je nepraktično svaki put pokretati cijelu skriptu. Na početku skripte potrebno je unijeti podatke o stroju koji se spremaju u strukturu *motor_data*: broj pari polova, broj utora, broj paralelnih grana, nazivna struja, nazivni napon, induktivitet glava namota, otpor glava namota, otpor faze, nazivni kut struje (prema MTPA) i frekvencija. Induktivitet i otpor glava namota moguće je postaviti u modelu i tada se u skripti zadaju

na nulu. Nazivni kut struje γ na fazorskom dijagramu 2.2. predstavlja negativan kut između induciranog napona na q-osi i struje statora. On određuje maksimalni moment u stroju i može se odrediti MTPA algoritmom (*eng. Maximum Torque Per Ampere*).



Slika 2.2. Fazorski dijagram IPM stroja

Struktura podataka *model_data* sadržava podatke vezane za modele za: mapiranje, kratki spoj i metodu sekante. Osim imena pojedinog modela i imena objekata u modelu, spremljeni su podaci o broju polova modela za mapiranje, trajanju simulacije kratkog spoja izraženog preko broja perioda te je spremljen podatak o vremenskom koraku simulacije kratkog spoja. Uz to, spremljeni su nazivi parametara u modelu za kratki spoj koji definiraju ponašanje električnog kruga. Podatak o broju polova u modelu bitan je za skaliranje ulančenih tokova. Faktor množenja k određuje granice struje u amperima za mapiranje, a početni kut rotora određen je tako da se u modelu os faze A poklapa s d-osi magneta rotora. Budući da u tropolnom kratkom spoju u stroju mogu poteći struje i do deset puta veće od nazivne, potrebno je imati mapu dovoljno širokih granica da se pokriju vrijednosti u prijelaznoj pojavi kratkog spoja. Korisnik može proizvoljno imenovati

datoteke za spremanje mapa. Imena svitaka i rotacijske komponente korisnik zadaje u modelu i imena unosi u skriptu. Prema tim imenima komponenata se dohvaćaju rezultati simulacija. Isto tako je proizvoljan izbor imena parametara u modelu za kratki spoj. Struktura podataka *file_name* sadržava imena datoteke s podacima o motoru i modelu, imena datoteka za direktnu i za inverznu mapu, ime datoteke za tranzijent kratkog spoja iz MAGNET-a i Eulerove metode te ime datoteke za rezultate metode sekante. Ovisno o procesorskim sposobnostima računala koje korisnik koristi, može se odabrati broj jezgara za paralelizaciju proračuna, preko varijable *n_cores*. Posljednji unos u ovoj sekciji koji se zahtijeva od korisnika jesu dimenzije mape. Varijabla *dimensions* je polje s dva člana: prvi član je broj točaka struje I_d, a drugi predstavlja broj točaka struje I_q. U slučaju kad bi se radile 3D mape stroja, varijabla bi imala i treću vrijednost koja bi se odnosila na broj pozicija rotora. Ovisno o stroju određuje se potrebna veličina mape. Preporučuje se da korak za struju I_d bude do 10% nazivne vrijednosti i do 20% za I_q. Na primjer, za stroj s nazivnom strujom 10,5 A, amplituda statorske struje iznosi 15 A te uz faktor $k = 7$ daje granicu od 105 A. Zato se preporučuje izraditi mapa (i_d^m, i_q^n), gdje su $m = 120$ i $n = 100$ veličine mape.

Sljedeća sekcija poziva funkciju za izradu direktnih mapa ulančenih tokova *direct_flux_map()* koja sama sprema mapu u strukturu imena *motor* te u pripadajuću datoteku. Budući da funkcija sprema podatke i nisu potrebne izlazne vrijednosti, izlaz funkcije je 0 ili 1, kao signalizacija da je mapiranje dovršeno. U slučaju korištenja 3D mapa može se uračunati skošenje i potom inverzna mapa. Podaci 3D mape se usrednje po poziciji i tako se dobiva 2D mapa čiji se inverz koristi u analitičkom proračunu tranzijenta kratkog spoja. Ako se odmah mapira 2D mapa, ona će biti puno detaljnija jer zahtijeva manje točaka za FEM analizu, ali ulančeni tokovi odnose se samo na nultu poziciju rotora. Na taj način podaci su ispravni, ali nepotpuni.

Iduća sekcija računa inverznu mapu d i q struja u ovisnosti o ulančenim tokovima. Ulaz funkcije je pokazivač na datoteku sa spremljenim podacima o stroju i modelu, a izlaz je inverzna mapa stroja *motor_inverse* koja se sprema u novu datoteku. Izračunate mape mogu se grafički prikazati, za lakšu vizualizaciju. Skripta *plot_map* prikazuje ovisnosti toka ψ_d i ψ_q o strujama I_d i I_q, prema direktnoj mapi. Unutar skripte izrađuju se mape napona koje se koriste za vizualizaciju i provjeru točnosti mapa ulančenih tokova.

Ako se u nazivnoj točki dobiva nazivni napon, mape su ispravne. Skripta još crta moment iz mape te se iz inverznih mapa crtaju ovisnosti struja I_d i I_q o ulančenim tokovima.

Slijede dvije sekcije za FEM simulaciju kratkog spoja i analitički izračun tranzijenta kratkog spoja Eulerovom metodom. U prvoj sekciji poziva se funkcija *short_circuit_simulation()* koja pokreće MAGNET model i dohvaća podatke o struji, toku i momentu u kratkom spoju. Podaci se spremaju u strukturu *motor_SC* koja se sprema u novu pripadajuću datoteku. Analitički proračun tranzijenta kratkog spoja radi funkcija *current_calculation_demag()* čiji je ulazni parametar pokazivač na datoteku s podacima, a izlaz je struktura podataka. Nju sačinjavaju podaci o maksimalnim iznosima struje I_d i struje statora I_s tijekom kratkog spoja, te polja vrijednosti struja i tokova u d i q osi, momenta te polje vremenskih koraka izvođenja proračuna. Struktura se sprema u varijablu *motor_demag* koja se koristi u idućoj skripti *plot_transient* za crtanje usporedbe FEM simulacije i analitičkog proračuna. Na kraju se poziva funkcija za metodu sekante *secant()* koja iterativnim proračunom određuje koja je struja I_d potrebna da se u statoru postigne tok magneta sa suprotnim predznakom. To je struja koja može uzrokovati demagnetizaciju. Rezultati metode sekante crtaju se skriptom *plot_secant*.

2.2.2. Direktna mapa stroja

Funkcija koja izrađuje direktne mape ulančenih tokova je *direct_flux_map()*. Funkcija raspodjeljuje proračun na više jezgri i uzastopno poziva pomoćnu funkciju za pokretanje simulacije u jednoj točki u Simcenter MAGNETU te nakon svakog koraka sprema dosad dobivene podatke u datoteku za direktnu mapu. Ulazni podatak funkcije je pokazivač na datoteku u kojoj su strukture *motor_data* s podacima o stroju i *model_data* s podacima o modelu te *file_name* s imenima datoteka u koje su strukture pohranjene. Prema pokazivaču na datoteku koji je predan kao ulazni podatak se učitavaju podaci iz datoteke. Ako direktna mapa već postoji i nije dovršena, varijabla *iteration* postavlja se na idući korak u izračunu mape i nastavlja s izradom mape. Ako datoteka s mapom ne postoji, *iteration* se postavlja na nulu te kreće mapiranje stroja. Varijabla *n_par* označava broj točaka za struju I_q, a varijablu *n_cores* zadaje korisnik, na koliko jezgri paralelno će se raditi proračun. Te dvije varijable određuju koliko će iteracija ukupno biti potrebno za mapirati cijeli stroj. Redom se uzimaju vrijednosti struje I_q, onoliko njih koliko je jezgri, te se za pojedinu struju pokreće simulacija za svaku vrijednost struje I_d.

Funkcija *parpool()* pokreće paralelni rad Matlaba. Varijabla *iteration* se u svakom koraku povećava za broj jezgara. Potrebno je uzeti u obzir situaciju dok *n_par* nije djeljiv s *n_cores* i broj iteracija, tj. *iteration* neće biti cijeli broj. Zato se u svakom koraku provjerava je li zbroj $iteration + n\_par$ veći od *n_par*. U slučaju u kojem jest, odnosno kad se u zadnjem koraku prijeđe preko zadanog broja točaka struje I_q, onda se varijabla *n_processes* postavlja na razliku $n\_par - iteration$. U suprotnom, tj. u svim prethodnim koracima se *n_processes* postavlja na *n_cores*. Petlja *for* čiji brojač ide od 1 do *n_processes* u svakom svojem koraku poziva funkciju *calc_part_new()*. Posebna Matlabova funcija *parfeval()* nudi mogućnost paralelnog poziva funkcije, u ovom slučaju poziva se funkcija *calc_part_new()*.

Funkcija *calc_part_new()* ima niz ulaznih parametara: indeks struje I_q za koje će se provesti FEM analiza, dimenzije mape, početno vrijeme simulacije i dvije strukture podataka s podacima o motoru i modelu. Indeksi struje zadaju se iz nadređene funkcije, ovisno o broju točaka struje u mapi. Dimenzije mape odredit će broj simulacija u jednoj iteraciji, a početno vrijeme služi za izračun trajanja jedne iteracije. Iz datoteke s podacima čitaju se i spremaju podaci o broju pari polova, broju polova u modelu, broju paralelnih grana, nazivnoj struji, faktoru množenja k, otporu i induktivitetu glava namota. Ako je model četvrtine stroja u MAGNET-u pripremljen za mapiranje, potrebno je preračunati iznose ulančenih tokova tako da se dobiju iznosi koji odgovaraju cijelom stroju. Stoga se uvodi varijabla omjera simuliranog toka *simulated_ratio* koja se računa prema jednadžbi 2.1

$$omjer\ simuliranog = \frac{ukupni\ broj\ polova}{broj\ polova\ u\ modelu\ \cdot broj\ paralelnih\ grana} \tag{2.1}$$

Nakon učitanih podataka, pokreće se aplikacija MAGNET u kojoj će se raditi simulacije. Funkcijom *actxserver* pokreće se MAGNET, a prema imenu modela iz strukture *model_data* otvara se model za mapiranje stroja. Prema uputama za skriptiranje [7] postavlja se vidljivost prozora, otvara se model, dohvaćaju se konstante i izgled modela. Dok se aplikacija pokreće, Matlab računa točke struja I_d i I_q kao pripremu za simulacije. Broj točaka za svaku struju definiran je dimenzijom mape, a iznosi struja definiraju se različito za I_d i I_q. Najprije se odredi maksimalna vrijednost $I_{peak} = I_n \cdot k\sqrt{2}$, gdje je k faktor množenja zadan u glavnoj skripti. Prva točka za struju I_d je $-I_{peak}$, a posljednja $0,1I_{peak}$.

Za struju I_q uzimaju se simetrične granice $(-0,7I_{peak}, 0,7I_{peak})$. Iskustveno se pokazalo da su te granice dovoljne za struju I_q. Za struju I_d potrebne su šire granice za negativne vrijednosti, a kako bi se koristio samo jedan faktor množenja, smanje se granice za struju I_q. Raspon između granica struja podijeljen je na jednake dijelove, prema broju točaka iz dimenzija mapa i stvoreno je polje vrijednosti struja u tom rasponu. Polja vrijednosti struja spremaju se u varijable *Id_vec* i *Iq_vec*.

U strukturi *motor* objedinjeni su podaci struja u *abc* i *dq* sustavima, toka u *dq* sustavu i momenta. Prije nego li se pokrene *for* petlja za izračun tokova, prealocira se memorija za tri 2D matrice za strukturu *motor*. Još je potrebno pripremiti varijablu *offset* za početni položaj rotora i varijablu *ac* koja predstavlja jedinični prostorni vektor prema 2.2

$$ac = e^{j\frac{2\pi}{3}} \tag{2.2}$$

Prije petlje postavlja se kut rotora θ. Naredbom *setMotionPositionAtStartup* u MAGNET-u se definira pozicija rotora, koji je predstavljen rotirajućom komponentom korisnički definiranog imena. Nakon toga, određuje se matrica inverzne Clarke-Park transformacije Ks_inv, prikazana u 2.3 U matrici je θ_r električni kut u radijanima.

$$Ks\_inv = \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & 1 \\ \cos(\theta_r - \frac{2\pi}{3}) & -\sin(\theta_r - \frac{2\pi}{3}) & 1 \\ \cos(\theta_r + \frac{2\pi}{3}) & -\sin(\theta_r + \frac{2\pi}{3}) & 1 \end{bmatrix} \tag{2.3}$$

Budući da je mapa definirana s dva parametra, *for* petlja jest jednostruka. Za zadani indeks struje I_q vrti se petlja po struji I_d. Kombinacija struja I_d i I_q se preračunava u *abc* sustav pomoću Ks_inv i dodatno se dijeli s $\sqrt{2}$ da se dobije efektivna vrijednost. Naredbom *setCoilCurrent* se iz Matlaba postavljaju u MAGNET-u efektivne vrijednosti faznih struja i uključuje se opcija *"CurrentOnAtTransientStart"* tako da odmah prva točka proračuna bude zadana radna točka. Fiksno se postavlja vrijeme trajanja simulacije na nulu i fiksni vremenski korak na 0,01 ms. Potom se pokreće tranzijentni 2D proračun s uključenom rotacijom, a simulira se samo prva točka u vremenu. Ista rješenja dobila bi se statičkim 2D proračunom ili tranzijentnim 2D proračunom bez rotacije, ali za te slučajeve potrebno je podesiti rubne uvjete na modelu, što je dodatni posao. Tranzijentni

2D proračun s rotacijom samostalno podesi rubne uvjete u zračnom rasporu što smanjuje posao u pripremi modela.

Kad je simulacija gotova dohvaćaju se rezultati: tok svake faze dohvaća se naredbom *getFluxLinkageThroughCoil*, a moment naredbom *getMagneticScalarTorqueOnMotionComponent*. S obzirom na to da se rezultati odnose samo na dio stroja u modelu, preračunavaju se. Ranije izračunatim omjerom simuliranja *simulated_ratio* množi se ulančeni tok faze i tome se dodaje umnožak odgovarajuće fazne struje i induktiviteta glave namota. U slučaju da je induktivitet već u modelu, varijabla *L_ew* je u Matlabu nula i nema dodatnog doprinosa glava namota. U suprotnom kad je model "čist" i bez utjecaja glava namota, tada se dodaje umnožak fazne struje i induktiviteta glave namota. Nakon izvlačenja iznosa faznih tokova i preračunavanja, dobivaju se tokovi u abc sustavu koji odgovaraju cijelom stroju. Oni se jednadžbom 2.4 preračunavaju u dq sustav.

$$\psi_{dq} = \frac{2}{3}(\psi_a + ac \cdot \psi_b + ac^2 \cdot \psi_c) \cdot e^{-j \cdot \theta_r} \tag{2.4}$$

Realni dio ψ_{dq} je ψ_d, a imaginarni dio je ψ_q. Oni se pohranjuju u strukturu *motor* uz podatke o struji. Moment se također preračunava na cijeli stroj omjerom $ukupni\ broj\ polova/broj\ polova\ u\ modelu$ te se sprema u strukturu. Da se dobije mapa stroja, koristi se funkcija *griddedInterpolant(X1, X2, V, 'spline')*. Parametri X1 i X2 su struje I_d i I_q, V je varijabla čija mapa se želi izraditi: tokovi ψ_d i ψ_q te moment T_{em}. Izlazi funkcije mapiranja šalju se na izlaze funkcije *calc_part_new()* i prosljeđuju se funkciji *direct_flux_map()* na daljnju obradu i spremanje.

Nakon paralelnog proračuna po pojedinoj struji I_q, funkcijom *fetchNext()* se dohvaćaju rezultati. U prvoj iteraciji spremaju se cjelokupne mape kako bi se alocirala memorija za ostatak podataka. Kasnija rješenja točno su indeksirana i prema indeksima nadopunjuju postojeću strukturu. Nakon spremanja rezultata završava se paralelni rad i briše se varijabla f u koju su spremljeni izlazi funkcije *parfeval()* jer će u idućoj iteraciji doći do neželjenog preklapanja varijabli. Struktura *motor* i varijabla *iteration* spremaju se u datoteku imena kojeg je korisnik zadao. U slučaju da je proračun prekinut zbog vanjskog utjecaja, skripta će učitati *iteration* i proračun će se neometano nastaviti gdje je zadnje spremljen. Za lakše praćenje napretka simulacije, pojavljuje se skočni prozor s napomenom koliko je iteracije obavljeno od ukupnog broja iteracija. Izlaz funk-

cije *direct_flux_map* je logička jedinica, u svrhu zastavice koja označava kraj proračuna. Glavna skripta će po tome znati da je izrada direktne mape ulančenih tokova završena i ispisati poruku o tome.

2.2.3. Inverzna mapa stroja

Funkcija za izradu inverzne mape stroja je *inverse_flux_map* i ima jedan ulazni parametar - pokazivač na datoteku sa spremljenim podacima. Nakon učitavanja datoteke s podacima o stroju i modelu, učitava se datoteka s direktnom mapom. Izrađuju se dvije vrste inverzne mape, s obzirom na upotrebu. Matlab ima dvije slične funkcije za mapiranje: *griddedInterpolant()* i *scatteredInterpolant()*. Strukturirana mapa s diskretnim vrijednostima koje su raspoređene u pravilnu rešetku (*eng. gridded*), kraće - pravilna mapa, ima drugačiji način interpolacije podataka od mape s diskretnim raštrkanim podacima (*eng. scattered*), skraćeno raštrkana mapa. Za tranzijent kratkog spoja bolje se pokazala raštrkana mapa. No, odmah će se izračunati pravilna inverzna mapa za potrebe grafičkog prikaza i za ostale potrebe kasnijeg proračuna.

Za izradu raštrkane mape bitne su kombinacije vrijednosti, ne njihov poredak. Uređene trojke (i_d, i_q, ψ_d), (i_d, i_q, ψ_q) i (i_d, i_q, T_{em}) dobivaju se izvlačenjem podataka iz direktnih mapa. Pomoćna struktura *data* ima spremljene vektore vrijednosti svake varijable. Uređenim trojkama se smatraju vrijednosti s istim indeksom u vektoru. U strukturu podataka *inv* spremaju se inverzne mape struje i momenta te se na kraju funkcije struktura prosljeđuje na izlaz funkcije. Za izradu pravilne mape potrebna je priprema drukčije mreže podataka. Veličina inverzne mape ostaje ista kao direktna mapa, što znači da će tok ψ_d imati jednak broj točaka unutar svojih granica kao i struja i_d. Analogno vrijedi za ψ_q i i_q. Iz direktnih mapa određuju se najveća i najmanja vrijednost toka u d i q osi. Prema krajnjim vrijednostima i zadanom broju točaka u dimenzijama mape, stvaraju se vektori vrijednosti tokova. Funkcijom *ndgrid()* stvaraju se 2D polja vrijednosti za ψ_d i ψ_q koja su temelj pravilne inverzne mape. Iz raštrkane mape određuju se vrijednosti struja za mrežu vrijednosti ψ_d i ψ_q. Funkcijom *griddedInterpolant()* izrađuju se mape struja, tokova i momenta u ovisnosti o tokovima ψ_d i ψ_q.

U slučaju da je izrađena 3D direktna mapa stroja, ona se može usrednjiti po poziciji. Time se dobiva 2D mapa čiji se inverz potom računa. Generalni nedostatak proračuna s

2D mapom je taj da se izgube efekti zuba, tj. valovitost ulančenog toka i momenta. Stoga će rezultati prema 3D direktnoj mapi biti točniji nego rezultati prema 2D direktnoj mapi.

2.2.4. Simulacija kratkog spoja

Jedini ulazni podatak funkcije *short_circuit_simulation()* je pokazivač na datoteku s podacima stroja. Nakon pokretanja aplikacije Simcenter MAGNET i otvaranja modela pripremljenog za simulaciju kratkog spoja, određuju se postavke simulacije. To su početno i krajnje vrijeme simulacije, vremenski korak te vektor s vremenskim trenucima za simulaciju. Naredbom *setFixedIntervalTimeSteps* postavljaju se zadana vremena u MAGNET-u. Naredbom *setParameter* postavljaju je parametri koji definiraju ponašanje električnog kruga. Redom su to trenutak uklopa sklopki, amplituda sinusne struje, fazni pomaci struje po fazama, frekvencija struje te induktivitet i otpor glava. Nakon što su svi parametri postavljeni, pokreće se tranzijentni 2D proračun s uključenom rotacijom.

Nakon dovršenoga proračuna dohvaćaju se rezultati. Napisane su tri pomoćne funkcije za čitanje polja podataka iz rezultata: *MNgetCurrent* za dohvaćanje struja, *MNgetFluxLinkage* za dohvaćanje ulančenih tokova i *getTorque* za dohvaćanje momenta. Struja se u *MNgetCurrent* dohvaća naredbom *getCurrentThroughCoil*, a tok se u funkciji *MNgetFluxLinkage* dohvaća naredbom *getFluxLinkageThroughCoil*. Moment se u funkciji *getTorque* dohvaća naredbom *getMagneticScalarTorqueOnMotionComponent*. Ulazni parametri funkcija su: interna varijabla preko koje je pokrenuta aplikacija MAGNET te ime svitka čija se struja i tok dohvaćaju. Odnosno, ime rotacijske komponente čiji se moment dohvaća.

Kao u funkciji *calc_part_new()*, definira se jedinični prostorni vektor i provodi se Clarke-Park transformacija iz abc sustava u dq prema 2.4 Analogna se formula može primijeniti na struje. Podaci struja i toka u abc i dq sustavu te moment stroja spremaju se u strukturu *motor_SC* koja se vraća kao izlaz funkcije *short_circuit_simulation()* te se u glavnoj skripti sprema u datoteku za kratki spoj prema FEM simulaciji.

2.2.5. Analitički proračun kratkog spoja

Kod analitičkog proračuna tranzijenta kratkog spoja koriste se inverzna i direktna mapa stroja. Na početku funkcije *current_calculation_demag()* dohvaćaju se podaci o

stroju koji su potrebni pri određivanju tranzijenta: otpor namota, frekvencija struja, broj polova, broj paralelnih grana, nazivna struja i kut struje γ. Računa se električna kutna brzina u radijanima i vršna vrijednost statorske struje. Iz strukture *model_data* očitavaju se podaci o trajanju simulacije kratkog spoja i vremenskom koraku simulacije. Prema frekvenciji statorskih struja izračuna se period te se podijeli s vremenskim korakom da se dobije broj točaka po periodi. Vektor s vremenskim trenucima generira se funkcijom *linspace()* od nule do kraja trajanja simulacije, s ukupno brojem točaka iz umnoška broja perioda i broja točaka po periodi. Dakle, na prvom mjestu vektora je nula, a na drugom je vremenski korak simulacije. Korak je korisnički zadan i isti je za FEM simulaciju i Eulerovu metodu. Podešavaju se početne vrijednosti struja, tokova i momenta. Struje se dobivaju prema jednadžbi 2.5, a tokovi i moment očitaju se iz direktnih mapa.

$$\begin{aligned} i_d &= \hat{I}_s \sin\gamma \\ i_q &= \hat{I}_s \cos\gamma \end{aligned} \tag{2.5}$$

Zadaju se prazni vektori za svaku komponentu struje i ulančenog toka te momenta i nakon toga se na prvo mjesto postavljaju početni uvjeti. Budući da je u kratkom spoju napon nula, ulančeni tok se mijenja u ovisnosti o struji. Za svaki vremenski korak trajanja simulacije računaju se Eulerove jednadžbe 3.2 Svaka promjena ulančenog toka uzrokuje promjenu struje i momenta, koji se čitaju iz inverznih mapa 3.3 Kad je *for* petlja završena, u vektoru struje i_d traži se najmanja vrijednost, tj. najnegativniji iznos koji je ključan za demagnetizaciju. U vektoru statorske struje i_s traži se maksimalna vrijednost te se prema indeksu traži odgovarajuća struja i_q. Uređena trojka se prosljeđuje na izlaz funkcije kao prvi dio strukture. Ostali dijelovi strukture redom su vektori struje, ulančenih tokova, momenta i vremena u tranzijentu kratkog spoja. Glavna skripta dohvaća rezultate funkcije u varijablu *motor_demag* i sprema u novu datoteku.

2.2.6. Metoda sekante

Na početku funkcije *secant()* učitava se datoteka s podacima te se čitaju podaci o broju paralelnih grana, broju polova u modelu i početnoj poziciji rotora. Potom se provode tri FEM simulacije. Prvom se određuje tok magneta tako da se u q-os postavi nazivna vrijednost struje statora, a u d-os struja na nula. Tok magneta je apsolutna vrijednost dobivenoga toka $\psi_{dq} = \psi_d + i\psi_q$ i sprema se u varijablu *psi_M*. Drugom i trećom simu-

lacijom određuju se prve dvije točke za metodu sekante. Za prvu točku se zadaju obje komponente struja na nulu, a u drugoj točki je $I_q = 0\,\mathrm{A}$ dok je struja I_d zadana na četvrtinu nazivne vrijednosti. Napisana je pomoćna funkcija *MagnetSecant()* koja provodi FEM simulaciju u MAGNET-u. Ulazni parametri *MagnetSecant()* su struje I_d i I_q, broj pari polova i paralelnih grana, omjer simuliranog, početna pozicija rotora te ime modela. Izlazi funkcije su tokovi ψ_d i ψ_q.

Ograničava se broj koraka metode sekante na 10, što se smatra dovoljno da se dosegne 1% tolerancije. Zadaju se prazni vektori struje I_d i toka ψ_d na čije se prvo i drugo mjesto postavljaju prve dvije izračunate točke. Na početku *for* petlje ispituje se u kojem je koraku metoda sekante. Ovisno je li u prvom, drugom ili daljnjem koraku postavlja se varijabla *psi_max* koja određuje mjesto presjeka. Nakon toga radi se interpolacija nove vrijednosti struje koja ulazi u FEM analizu, iz koje se dobiva nova vrijednost toka. Kad je dobivena vrijednost toka za 1% različita od toka magneta *psi_M*, *for* petlja završava s izvođenjem i podaci se prosljeđuju na izlaz funkcije. Detaljniji opis metode sekante uz grafički prikaz je u poglavlju 4.

2.2.7. Grafički prikazi

Tri su skripte napisane za izradu grafičkih prikaza rezultata. Prva je *plot_map* za prikaz mapa, druga je *plot_transient* za prikaz rezultata proračuna kratkog spoja, a treća je *plot_secant* koja iscrtava trend rješavanja metodom sekante.

Na početku skripte za crtanje mapa učitavaju se datoteke s podacima o stroju te direktnim i inverznim mapama stroja. U prvoj sekciji se funkcijom *surf()* prikazuju ovisnosti ulančenih tokova o strujama I_d i I_q. U drugoj sekciji se prvo računaju mape napona koje se potom crtaju u ovisnosti o strujama I_d i I_q. U trećoj se sekciji crta mapa napona te se u posljednjoj sekciji crtaju inverzne mape struja I_d i I_q u ovisnosti o tokovima ψ_d i ψ_q. Grafički prikazi služe da se provjeri jesu li podaci u mapama potpuni i odgovarajućih iznosa te jesu li mape glatke.

Skripta *plot_transient* služi za usporedbu rezultata dobivenih FEM simulacijom i analitički. Iz tog se razloga moraju učitati datoteke s podacima o kratkom spoju iz MAGNET-a i analitičkom Eulerovom postupku određivanja tranzijenta kratkog spoja. Vrijednosti struja, tokova i momenta iz struktura se pridjeljuju varijablama kratkih imena radi jed-

nostavnosti daljnjeg koda. Dodatno, izračunaju se ukupna struja prema $I_s = \sqrt{I_d^2 + I_q^2}$ i ukupni tok $\psi_s = \sqrt{\psi_d^2 + \psi_q^2}$. Vektori vrijednosti struja, tokova i momenta iz FEM-a imaju dio podataka prije trenutka kratkog spoja. Ako se poznaje trenutak kad sklopke u FEM modelu uklope i izazovu kratki spoj, može se izračunati vrijednost varijable *moving*, koliko je vremenskih koraka spremljeno prije uklopa. Funkcijom *circshift()* se radi kružni pomak podataka u vektoru. Svim vektorima podataka iz FEM-a se rotiraju vrijednosti tako da dođu na kraj vektora. Pri crtanju se taj višak u vektorima odreže i prikazuje se samo dio relevantan za kratki spoj. Prikazuje se sedam grafova koji uspoređuju rezultate iz FEM-a i analitike. Na prva dva grafa uspoređuju se struje po komponentama d i q te ukupna struja I_s. Na trećem i četvrtom grafu uspoređuju se d i q komponente ulančenog toka te ukupni ulančeni tok ψ_s. Peti graf prikazuje usporedbu struja, a šesti graf prikazuje ulančane tokove u kompleksnoj ravnini. Taj način prikaza omogućuje drugačiju predodžbu trenda promjene tijekom tranzijenta kratkog spoja. Posljednji, sedmi graf prikazuje usporedbu momenata tijekom kratkog spoja.

Posljednja skripta za grafički prikaz je *plot_secant* koja uzima vektore struje i ulančenog toka, koji sadrže najviše 12 točaka: dvije su početni uvjeti, a ostale su dobivene metodom sekante. Za glatko nacrtani graf potrebno je puno više točaka, radi toga se podaci interpoliraju na gustoj mreži. Točke iz metode sekante jasno su naznačene na grafu. Da se dodatno prikaže odnos toka magneta i ulančenog toka u tranzijentu kratkog spoja, ponovno se crta graf ulančenih tokova u kompleksnoj ravnini te se u ovom crtanju dodaje točka $(-\psi_M, 0)$.

Svaka od skripti će svoje grafove proslijediti glavnoj skripti, u kojoj će ti grafovi biti nacrtani. Tako su na jednom mjestu objedinjeni grafički prikazi rezultata proračuna, pogodni korisniku za usporedbu i analizu.

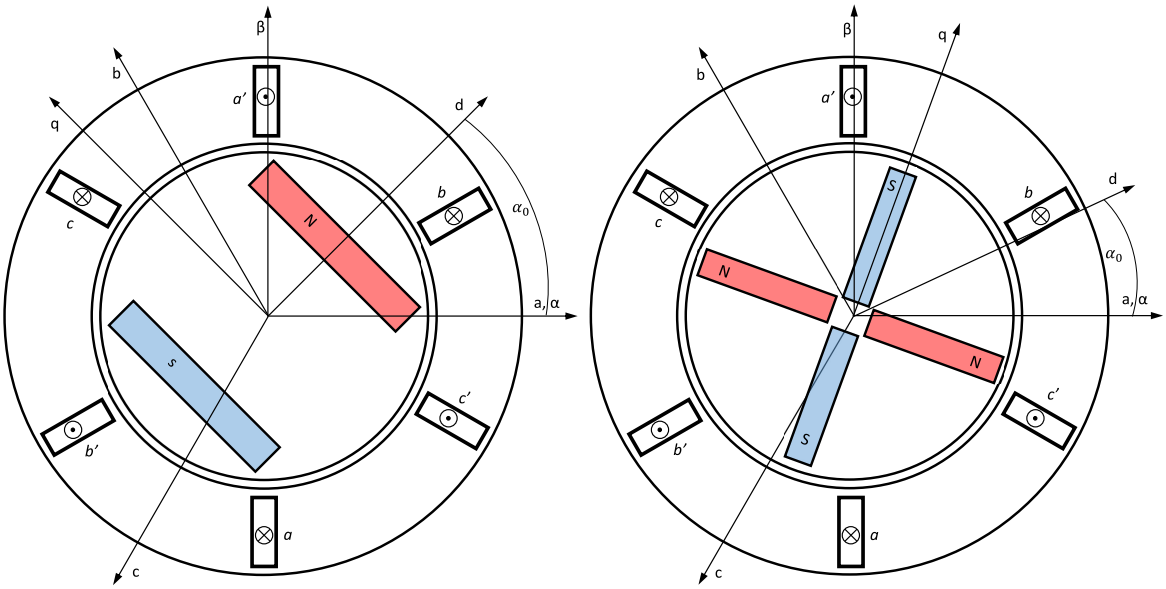
2.3. Priprema modela u Simcenter MAGNET-u

Skripte u Matlabu napravljene su prema određenim pretpostavkama. Kod pozivanja i provođenja simulacije u MAGNET-u moraju biti ispunjeni odgovarajući uvjeti kako bi proračun bio uspješan. Potrebna su dva modela: za mapiranje i za simulaciju kratkog spoja. Za proračun metode sekante koristi se model za mapiranje.

2.3.1. Model stroja za mapiranje

Model za mapiranje stroja trebao bi biti što jednostavniji, odnosno s najmanje moguće polova. Dovoljno je modelirati jedan prapol stroja jer se magnetska slika na ostatku stroja ponavlja. U glavnoj skripti se navede broj polova u modelu i time će se ulančeni tokovi točno preračunati. Poprečni presjek stroja može se nacrtati u bilo kojem alatu za crtanje, npr. AutoCAD i izvesti u *.dxf* formatu, koji je moguće učitati u MAGNET. Na temelju poprečnog presjeka definiraju se objekti zadane duljine i zadanog materijala. Veličine elemenata u mreži konačnih elemenata određuju se iskustveno: 0,5 mm u zračnom rasporu, 1,0 mm u zubu statora i magnetima, 2,0 mm u namotu, jarmu statora i rotora. Navedene vrijednosti primjerene su za manje električne strojeve. Za veliki sinkroni generator s permanentnim magnetima, na primjer, mogu se uzeti veći elementi za izradu mreže. Bitno je da mreža ima što više jednakostraničnih elemenata i da gustoća mreže bude proporcionalna gustoći magnetskog toka u tom dijelu stroja.

Namot je potrebno izraditi vodeći računa o smjeru struje u svakom utoru te svitke po fazama proizvoljno imenovati. Sve komponente mogu imati proizvoljna imena, u glavnu skriptu se unose potrebna imena za dohvat rezultata simulacije. Potrebno je definirati rotacijsku komponentu i definirati brzinu. Pri definiranju materijala magneta važno je voditi računa o usmjerenju magnetizacije. Ovisno o smjeru magneta koji se nalazi u modelu, određuje se početni kut rotora α_0. Na početku svake simulacije moraju biti poravnate osi faze A i d-os rotora. Na slici 2.3. su prikazani koordinatni sustavi za dvije različite topologije strojeva s unutrašnjim permanentnim magnetima. Zbog crtanja poprečnog presjeka i tek kasnije izrade namota, dogodi se da osi u modelu nisu poravnate, kako je prikazano na slici. Kut α_0 mora biti uzet u obzir kako bi se osi poravnale. Poravnanje osi može se osigurati na dva načina. Postavljanjem početne pozicije u postavke rotirajuće komponente rotor se namješta u odgovarajuću poziciju i kut α_0 jednak je nuli. Varijabla u skriptama koja opisuje geometrijski pomak iznosa α_0 je *offset*. Druga opcija je uračunavanje α_0 u kut statorskih struja. U ovom radu korišten je pristup postavljanja početne pozicije, no jednako točno je postaviti jednadžbe statorskih struja po fazama prema 2.6 Nakon pretvorbe iz abc u dq sustav s uračunatim α_0, os statora će se pomaknuti i poravnati s rotorom. Kod sinkronih strojeva s permanentnim magnetima nazivni kut struje određuje se prema MTPA ili nekim drugim optimizacijskim postupkom. Na-

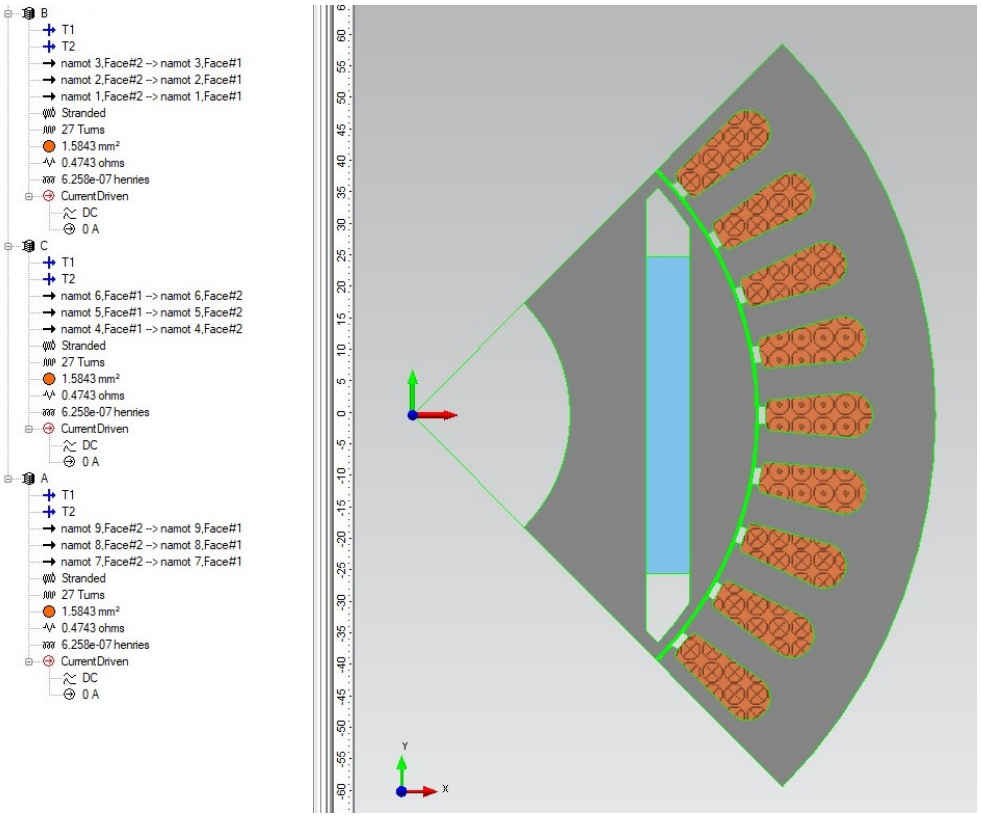


Slika 2.3. Prikaz abc, $\alpha\beta$ i dq koordinatnih sustava na dvije topologije motora s unutrašnjim permanentnim magnetima

zivni kut je γ i predstavlja negativni fazni pomak struje u odnosu na inducirani napon, prema slici 2.2. Ukupni fazni pomak struje je $\phi_0 = \pi - \gamma - \alpha_0$.

$$\begin{aligned} i_a(t) &= \hat{I}_s \ \sin(2\pi f_s \cdot t + \phi_0) \\ i_b(t) &= \hat{I}_s \ \sin(2\pi f_s \cdot t + \phi_0 + \frac{2\pi}{3}) \\ i_c(t) &= \hat{I}_s \ \sin(2\pi f_s \cdot t + \phi_0 - \frac{2\pi}{3}) \end{aligned} \tag{2.6}$$

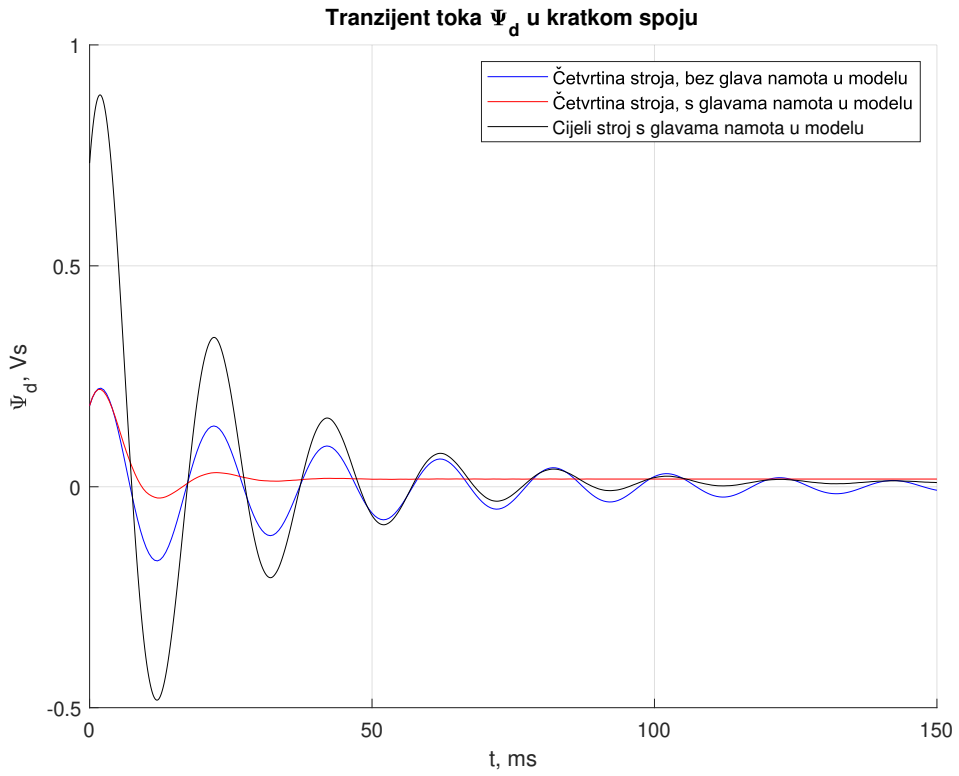
Za mapiranje stroja, dovoljna je 2D FEM simulacija, no potrebno je dodati efekte 3D modela, koji uključuju utjecaje glava namota. Induktivitet i otpor glava namota poznati su iz proizvođačevih specifikacija stroja. Uračunati se mogu na dva načina: upisati u glavnu skriptu u Matlabu ili unijeti kao svojstvo svitka u modelu. U modelu za mapiranje stroja, utjecaj glava namota mora se unijeti kao svojstvo svitka, kako je prikazano na slici 2.4., jer se električni krug ne koristi za 2D simulaciju u jednoj točki.



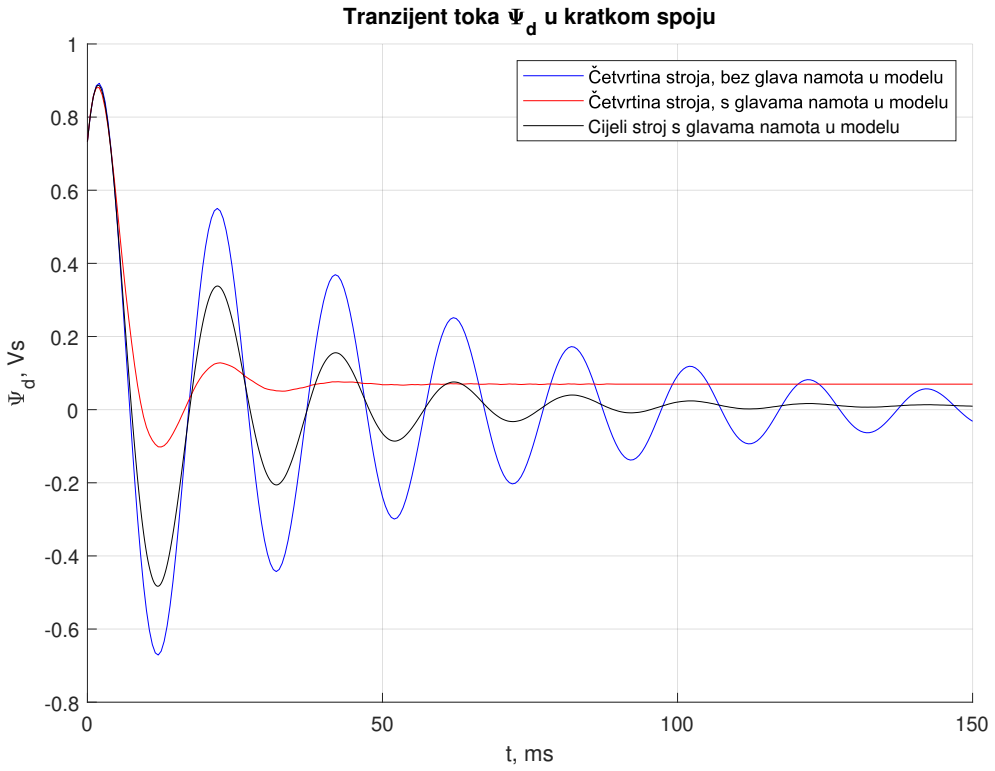
Slika 2.4. Induktivitet i otpor glava namota kao svojstvo svitka u modelu

2.3.2. Model stroja za simulaciju kratkog spoja

Analogno crtanju modela za mapiranje, potrebno je provesti korake u crtanju modela za kratki spoj. No, u slučaju kratkog spoja potrebno je modelirati cijeli stroj. Ako se modelira samo dio stroja, FEM simulacija izvrši kratki spoj gdje se efekti glava namota ne uračunaju ispravno. Slika 2.5. prikazuje usporedbu u odzivima ulančenog toka ako su modeli različito definirani. Prvi odziv plave boje prikazuje odziv toka ψ_d za model bez otpora i induktiviteta glava namota, a drugi koji je crvene boje ima modelirane i parametre glava namota. To su modeli četvrtine četveropolnog stroja, dakle modeliran je samo jedan pol. Treći odziv je crne boje, od modela cijelog stroja s modeliranim otporom i induktivitetom glava namota. On je u početnom trenutku četiri puta većeg iznosa od prva dva, razlog tome je razlika u veličini modela. No, ako se prva dva odziva pomnože s omjerom simuliranog koji iznosi četiri, to ne rješava problem, kako prikazuje slika 2.6.



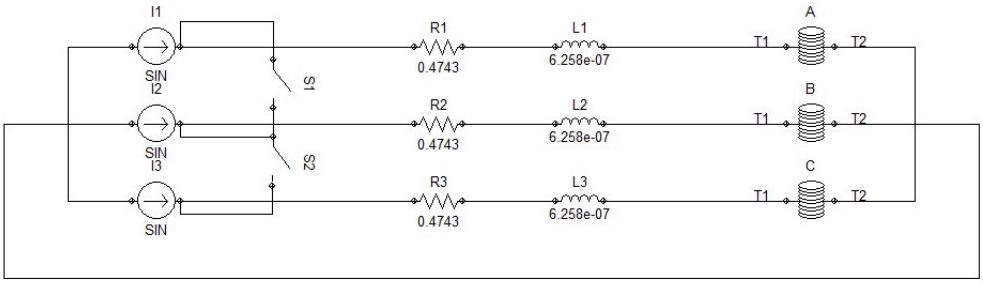
Slika 2.5. Usporedba tranzijenta toka u *d*-osi za različito definirane modele, s originalnim odzivima FEM simulacije



Slika 2.6. Usporedba tranzijenta toka u *d*-osi za različito definirane modele, skalirani tako da se izjednači početna vrijednost

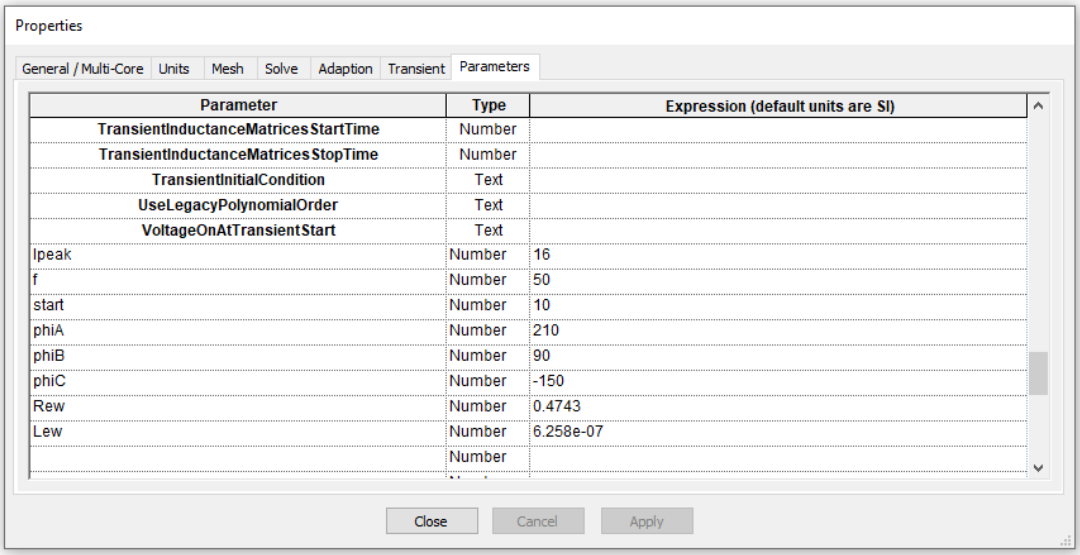
Prikazani rezultati su napravljeni da se pokaže opravdanost zahtjeva cjelovitog modela za simulaciju kratkog spoja.

Nakon što se definira geometrija stroja, aksijalna duljina modela i materijali svakog objekta, potrebno je definirati svitke. Ako stroj ima paralelne grane, one se mogu modelirati kao zasebni svici te se u električnom krugu spoje na odgovarajući način. Na slici 2.7. je prikazan slučaj bez paralelnih grana. Uz to, prikazan je drugi način uključivanja efekata glava namota u model - tako da se otpor i induktivitet postave u električni krug. Glavna razlika u modelu za mapiranje i za kratki spoj jesu sklopke. Njima se zadaje vrijeme kada će uklopiti i time izazvati trajni tropolni kratki spoj. Trenutak u kojem se sklopke uklapaju mora biti dobro izabran, zbog pozicije rotora u odnosu na stator.



Slika 2.7. Induktivitet i otpor glava u električnom krugu u modelu

Nije potrebno podešavati postavke simulacije jer će to skripta u Matlabu automatski podesiti. No, potrebno je definirati niz parametara koji upravljaju svojstvima električnog kruga. Na slici 2.8. prikazan je popis potrebnih parametara za električni krug. Potrebni su parametri za amplitudu struje, fazni pomak svake faze, frekvencija struje, trenutak za uklapanje sklopki te induktivitet i otpor glava namota. Ovo je samo primjer imenovanja parametara, izbor naziva je proizvoljan i korisnik odabrane nazive parametara unosi u glavnu skriptu i na taj način povezuje zadavanje podataka s modelom. Vrijednosti parametara zadaju se u funkciji *short_circuit_simulation()* čime se sprječava direktno zadavanje brojčanih vrijednosti u model.



Slika 2.8. Primjer imenovanja parametara u modelu za simulaciju kratkog spoja

U postavkama elemenata električnog kruga vrijednosti se unose parametarski. Primjer je prikazan za postavke strujnog izvora na slici 2.9.

Slika 2.9. Parametarsko zadavanje vrijednosti sinusnog strujnog izvora

3. Tranzijent kratkog spoja

U trenutku tropolnog kratkog spoja na stroju, napon na stezaljkama trenutačno pada na nulu, a statorske struje rastu. Ograničavaju ih otpor i induktivitet namota, pri čemu veliku ulogu igraju otpor i induktivitet glava namota. Prijelazne pojave koje prolaze struje i ulančeni tokovi nazivaju se zajedničkim imenom tranzijent kratkog spoja i glavni je predmet razmatranja ovog poglavlja. Tranzijent kratkog spoja vrijedan je pokazatelj ponašanja stroja tijekom kvara i potrebno ga je odrediti i analizirati prije proizvodnje i prodaje stroja. Mnogi programski alati dostupni su za određivanje tranzijenta kratkog spoja. Najpopularniji simulacijski alati temeljeni na FEM analizi su ANSYS, COMSOL Multyphisics, JMAG i Simcenter MAGNET. Od analitičkih alata na raspolaganju su Matlab, Octave i WolframAlpha te programski jezici koji imaju ODE pakete (*eng. Ordinary Differential Equations*) poput Pythona.

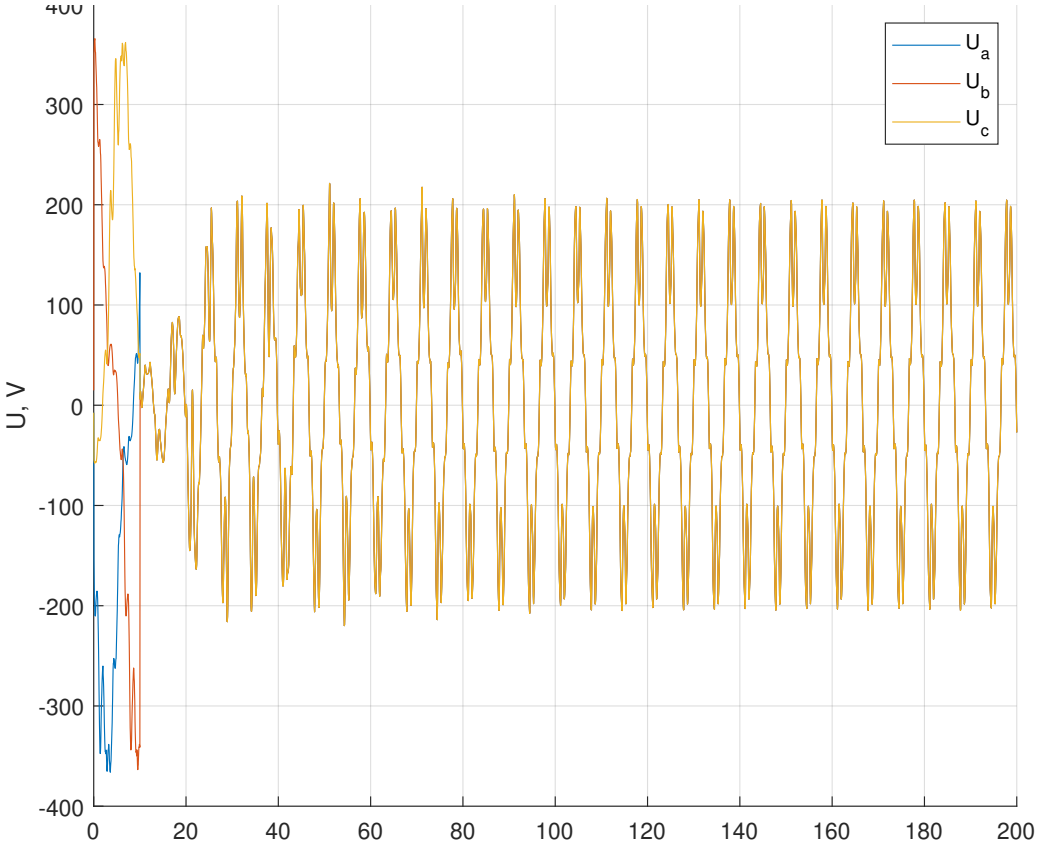
U nastavku su predložene i uspoređene tri metode, opisane su njihove prednosti i mane te su dane preporuke za korištenje. Jedna metoda je simulacijska - FEM analiza u Simcenter MAGNET-u, a ostale dvije su analitičke metode: jedna se temelji na Eulerovoj metodi u Matlabu, a druga na Simulink blokovskoj shemi. Preduvjet za analitički proračun je poznavanje mapa ulančenih tokova. Ideja za kodiranje Eulerove metode zasnovana je na programskom paketu SyR-e. Profesori Sveučilišta u Torinu razvili su Matlab aplikaciju za analizu električnih sinkronih strojeva s permanentnim magnetima i reluktantnim strojevima te cjelokupni kod objedinili u SyR-eu. Cilj je pokazati da se tranzijent kratkog spoja izračunat analitički unutar nekoliko sekundi može mjeriti s FEM proračunom koji traje više sati. Proračun tranzijenta u MotorCAD-u također traje kratko, no netočan je. Rezultati tog proračuna dani su u zasebnom poglavlju.

3.1. FEM simulacija kratkog spoja

Metoda konačnih elemenata (*eng. FEM - Finite Element Method*) numerički je postupak rješavanja diferencijalnih jednadžbi. Najčešće se primjenjuje u matematičkom modeliranju i inženjerstvu za rješavanje problema u mehanici, termodinamici, aerodinamici, hidrodinamici i elektrotehnici. Za rješavanje problema prvo se definira domena problema te rubni uvjeti te domene. Nakon toga, model se podijeli na pravilno raspoređene trokute ili tetraedre, ovisno je li problem u 2D ili 3D domeni. Svi elementi zajedno čine mrežu konačnih elemenata nad kojom su definirane interpolacijske funkcije. Za svaki element se definira elementarna matrica prema aproksimiranim jednadžbama polja. Koristeći elementarne matrice, sastavlja se linearni sustav matričnih jednadžbi čijim se rješavanjem dolazi do rješenja polja u domeni.[11]

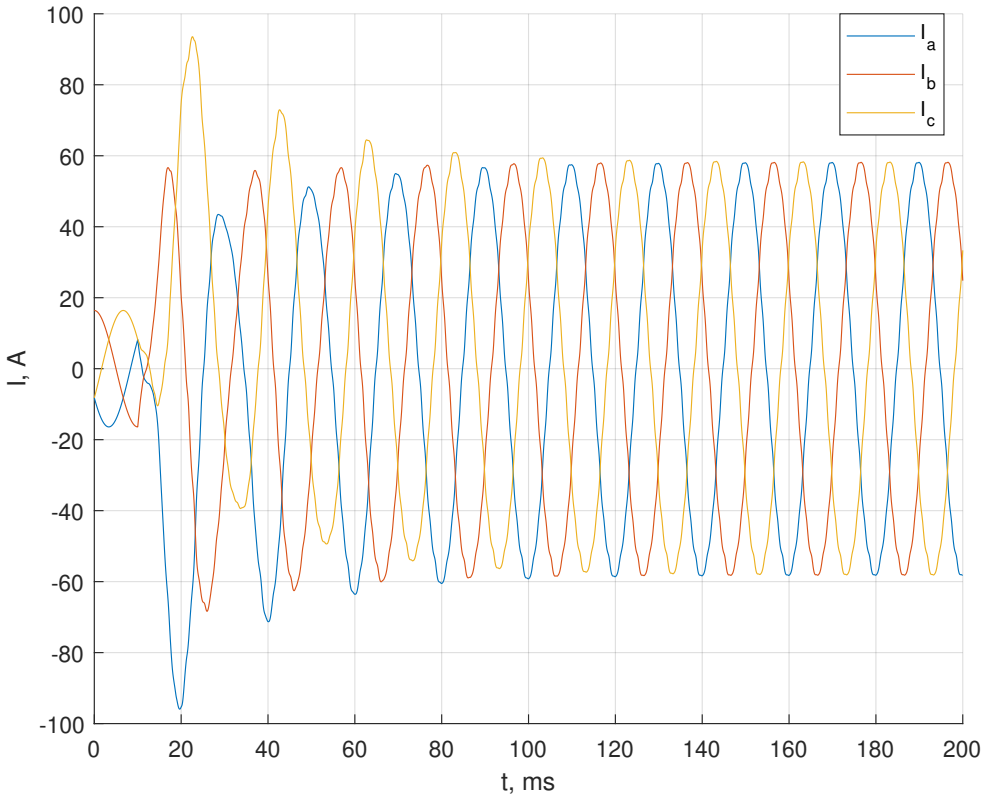
Upute za izradu modela za kratki spoj objašnjene su u poglavlju 2.3.2. Funkcija *short_circuit_simulation()* podešava parametre početka i trajanja simulacije te vremenskog koraka. Elementi električnog kruga definirani su parametarski, tako da funkcija može postaviti željene vrijednosti i pokrenuti 2D tranzijentnu simulaciju s uključenim gibanjem. Rezultati simulacije prikazani su na sljedećim slikama.

Prvih 10 ms stroj je u nazivnoj radnoj točki i tada nastupa kratki spoj, tj. sklopke u električnom krugu se uklope. U tom trenutku napon na stezaljkama pada na nulu, ali u namotu i dalje postoji inducirani napon zbog vrtnje rotora. Slika 3.1. prikazuje prijelaznu pojavu faznog napona. U desetoj milisekundi nastupa kratki spoj i napon sve tri faze se izjednačava.

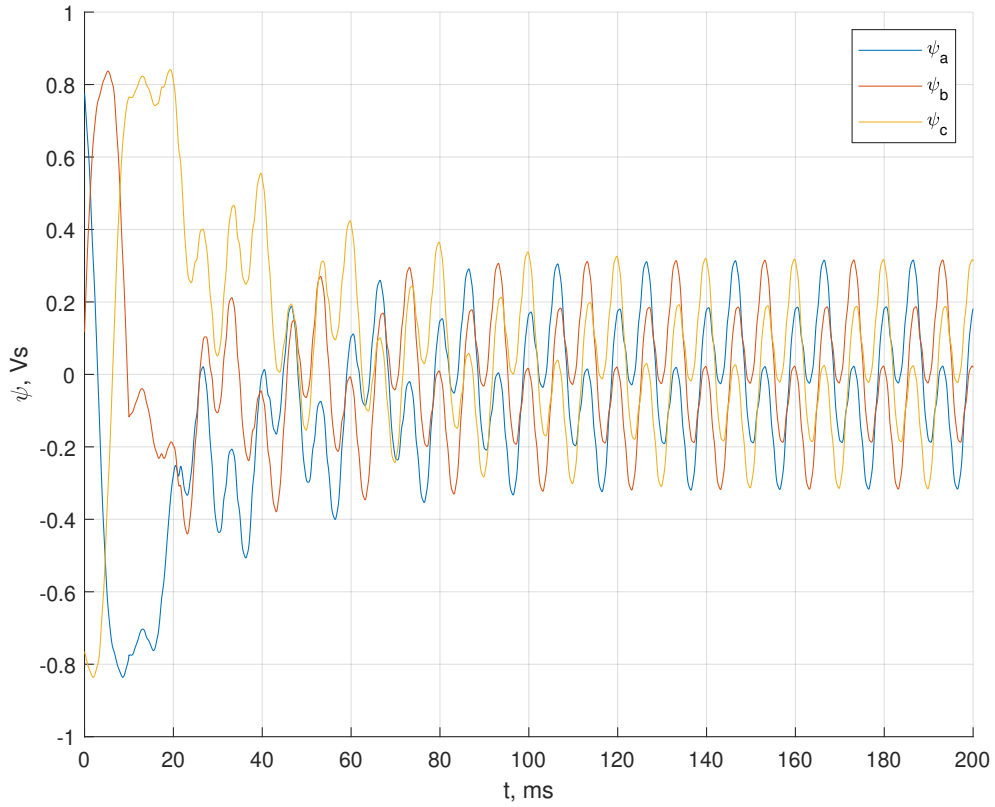


Slika 3.1. Valni oblik faznih napona u kratkom spoju

Na slici 3.2. je valni oblik faznih struja. Struje imaju veliko nadvišenje i nakon prijelazne pojave ulaze u stacionarno stanje za trajni kratki spoj. Prijelazna pojava traje 150 ms i ona se bolje vidi na grafu struja I_d i I_q. Fazni ulančeni tokovi imaju prijelaznu pojavu jednakog trajanja, no stacionarno stanje je drugačije. Slika 3.3. prikazuje fazne ulančene tokove u kratkom spoju. Struje u kratkom spoju ostaju oblika sličnog sinusu, a ulančeni tokovi se zbog viših harmonika izobličavaju. No, zbog periodičnosti valnog oblika, vrijednosti ulančenih tokova u dq sustavu jesu stacionarne. Struje i ulančeni tokovi u dq sustavu se prikazuju u poglavlju 3.4., gdje se uspoređuju rezultati FEM simulacije i Eulerove analitičke metode.



Slika 3.2. Valni oblik faznih struja u kratkom spoju



Slika 3.3. Valni oblik faznih ulančenih tokova u kratkom spoju

3.2. Eulerova analitička metoda određivanja tranzijenta kratkog spoja

Valni oblici struja i ulančenih tokova u kratkom spoju dobivaju se rješavanjem diskretiziranih naponskih jednadžbi. Brzina rotora uzima se konstantna, a gubici u bakru i željezu su zanemareni. Naponske jednadžbe 3.1 su diskretizirane Eulerovom metodom, po kojoj je cijela metoda za određivanje tranzijenta dobila ime.

$$\begin{aligned} u_d &= R_s \cdot i_d - \omega \cdot \psi_q + \frac{d\psi_d}{dt} \\ u_q &= R_s \cdot i_q + \omega \cdot \psi_d + \frac{d\psi_q}{dt} \end{aligned} \tag{3.1}$$

U kratkom spoju napon je nula te jednadžbe poprimaju oblik 3.2 Diskretni vremenski korak je Δt, a koeficijenti $k-1$ i k predstavljaju redom prošli i sadašnji vremenski diskretni trenutak. U toj je formi jednadžba implementirana u funkciji *current_calculation_demag*. Trenutni se tok računa iz prethodnih vrijednosti struja i tokova.

$$\begin{aligned} \psi_d^k &= (-R_s \cdot i_d^{k-1} + \omega \cdot \psi_q^{k-1}) \cdot \Delta t + \psi_d^{k-1} \\ \psi_q^k &= (-R_s \cdot i_q^{k-1} - \omega \cdot \psi_d^{k-1}) \cdot \Delta t + \psi_q^{k-1} \end{aligned} \tag{3.2}$$

Trenutni iznosi struje dohvaćaju se iz inverznih mapa 3.3 prema trenutnim vrijednostima tokova. Uz struje, dohvaća se trenutni moment iz inverzne mape momenta. Inverzne mape struja i momenta ranije su izračunate. Moment se može dohvaćati i iz direktnih mapa prema trenutnim vrijednostima struja, rezultati su isti.

$$\begin{aligned} i_d^k &= \mathbf{I_d}(\psi_d^k, \psi_q^k) \\ i_q^k &= \mathbf{I_q}(\psi_d^k, \psi_q^k) \\ T_{em}^k &= \mathbf{T_{em}}(\psi_d^k, \psi_q^k) \end{aligned} \tag{3.3}$$

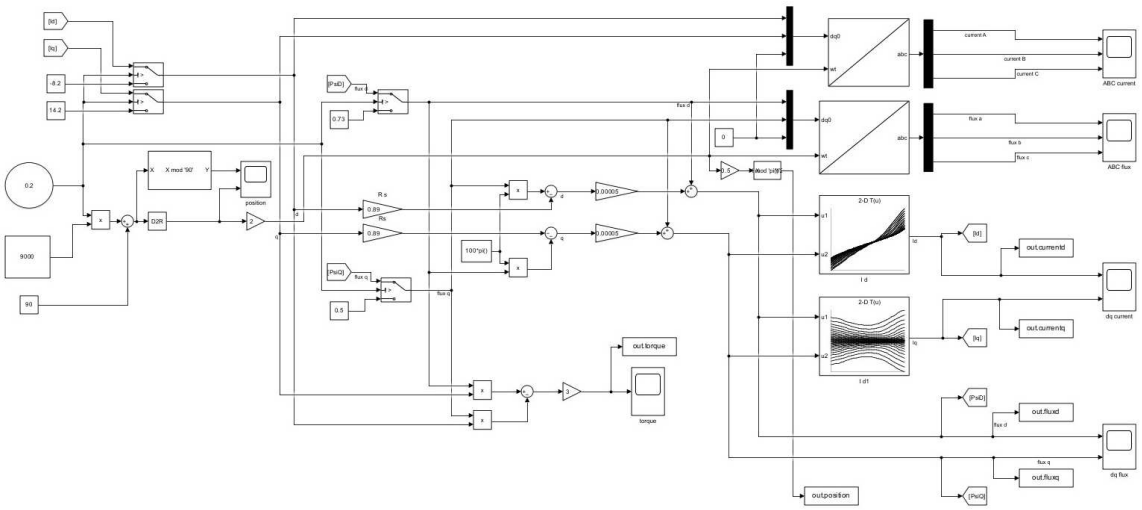
Proračun se ponavlja za sve vremenske trenutke tranzijenta, čije se trajanje mjeri u broju električnih perioda. Ranije se definira broj električnih perioda, kao trajanje proračuna kratkog spoja. Početni trenutak kratkog spoja uvelike određuje tranzijent. Za IPM stroj odabrana je radna točka prema MTPA i kut struje je $gamma = -30°$. To je ujedno najgori slučaj kratkog spoja, jer je moment najveći.

Ovaj postupak nema integriranu FEM simulaciju ni u jednom koraku, ali zahtijeva poznavanje mapa ulančenih tokova i njihov inverz. Mape se mogu dobiti nizom FEM simulacije ili mjerenjem na laboratorijskom postavu. U ovoj verziji proračuna ne uzima se u obzir položaj rotora jer se koriste 2D mape. Pretpostavlja se da su razlike u toku za svaku poziciju dovoljno male da ne budu od velikog značaja. Izračun mapa može trajati od nekoliko sati do nekoliko dana, ovisno o dimenzijama mapa. Eulerova metoda za određivanje tranzijenta kratkog spoja preko mapa ulančenih tokova traje svega nekoliko sekundi. Za proračun IPM motora potrebne su 2s, na računalu s AMD EPYC 7713 64-jezgrenim procesorom.

3.3. Određivanje tranzijenta kratkog spoja u Simulinku

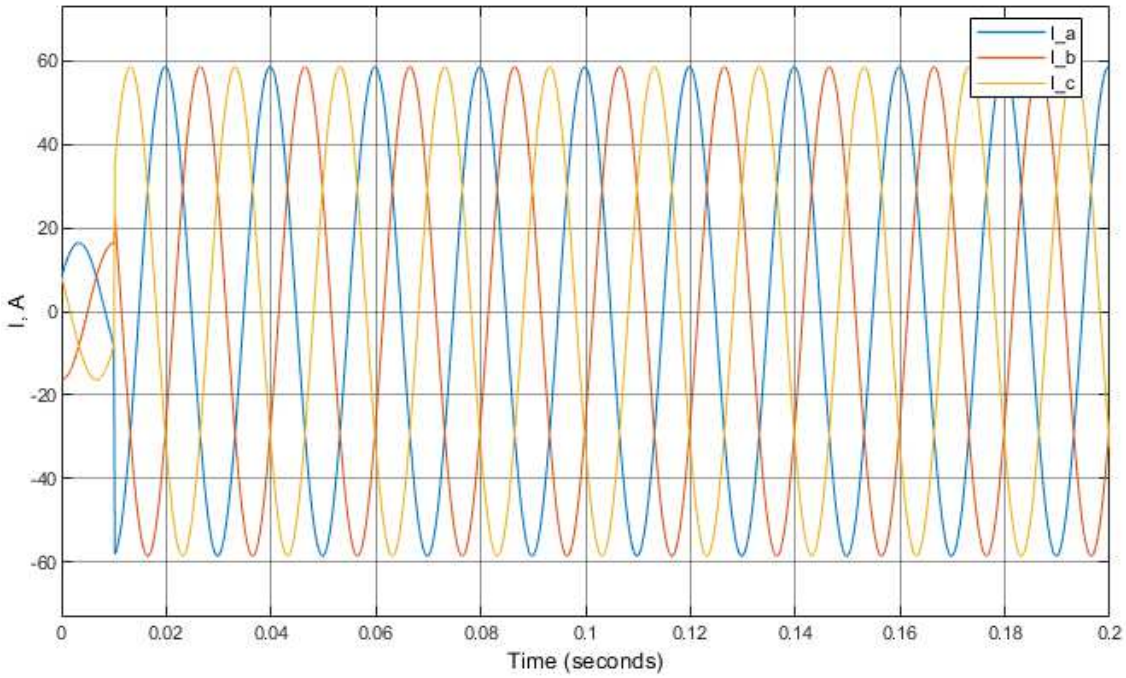
Simulink je alat unutar Matlaba u kojem se grafički programira i služi za rješavanje dinamičkih modela. Osim obrade signala, Simulink uključuje pakete za rješavanje problema u elektrotehnici, mehanici, hidraulici i termodinamici te cijeli niz paketa za integrirane sustave.

Model za analizu kratkog spoja izrađen je po uzoru Eulerove metode i prikazan je na slici 3.4. Ulazi u sustav su nazivne vrijednosti struja I_d i I_q i početna pozicija rotora kao početni uvjeti. U postavkama simulacije definiran je konstantni vremenski korak izvođenja i zadano je vrijeme izvođenja.



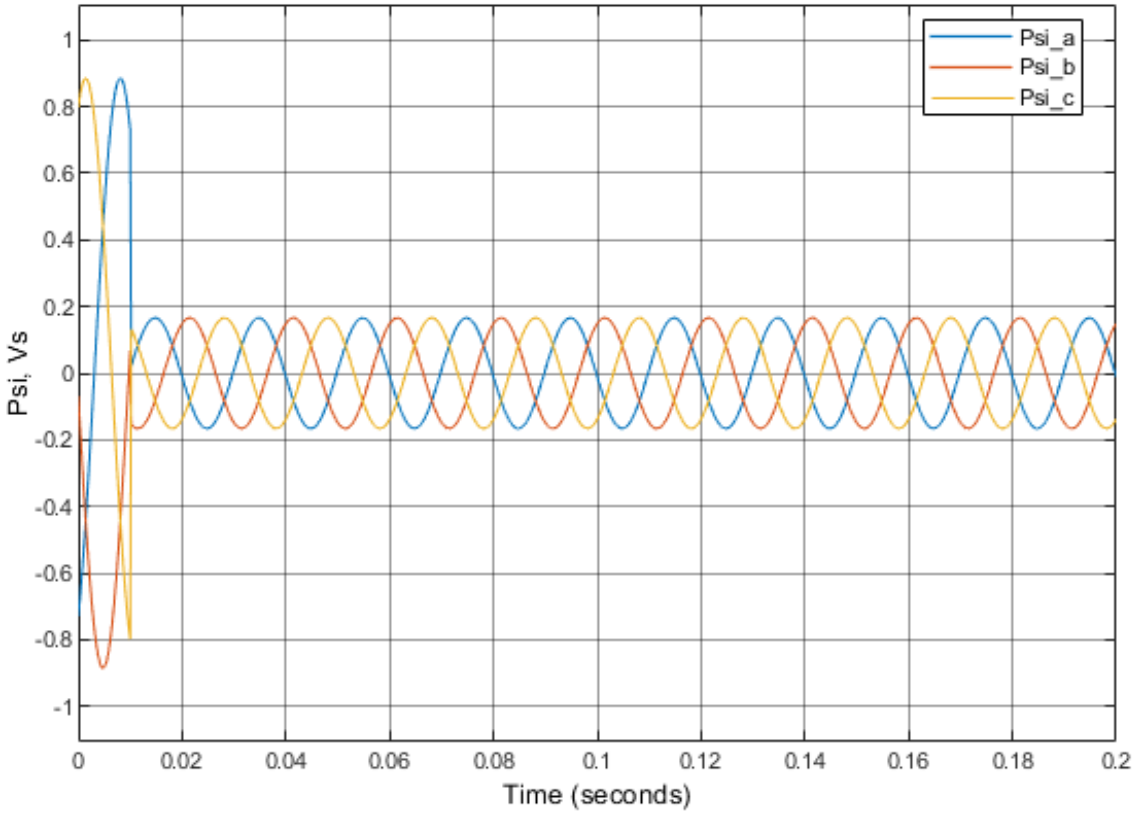
Slika 3.4. Blokovska shema za analizu tranzijenta kratkog spoja pomoću mapa

Da se izazove kratki spoj u modelu, korištene su sklopke koje u desetoj milisekundi, jednako kao i FEM simulacija, prebacuju sa zadanih nazivnih vrijednosti struja na vrijednosti koje se dohvaćaju iz inverznih strujnih mapa. Ulančeni tokovi se u trenutnom vremenskom koraku računaju s vrijednostima struja i ulančenih tokova iz prethodnog vremenskog trenutka. Oni se prosljeđuju na blok *LUT* (*eng. Look-Up Table*) u kojem su definirane inverzne mape stroja te su izlazi blokova struje I_d i I_q. Pozicija rotora računa se prema proteklom vremenu simulacije i prosljeđuje na blok za preračunavanje dq sustava u abc sustav, kako bi se dobile fazne vrijednosti struja i ulančenih tokova. Na slici 3.5. prikazani su valni oblici faznih struja prema analizi kratkog spoja u Simulinku.



Slika 3.5. Valni oblik faznih struja u tropolnom kratkom spoju, analiza u Simulinku

Jasno se vidi trenutak kratkog spoja, no nema prijelazne pojave. Slika 3.6. prikazuje valni oblik faznih ulančenih tokova iz analize u Simulinku. Slično kao kod struja, nema prijelazne pojave. Valni oblik prije kratkog spoja odgovara nazivnom radu, a nakon kratkog spoja odgovara stacionarnom stanju kratkog spoja. U potpoglavlju 3.5. prikazana je usporedba FEM simulacije i analize u Simulinku. Tamo su uspoređene struje i ulančeni tokovi u dq sustavu i prikazano je poklapanje valnih oblika u stacionarnom stanju.



Slika 3.6. Valni oblik faznih ulančenih tokova u tropolnom kratkom spoju, analiza u Simulinku

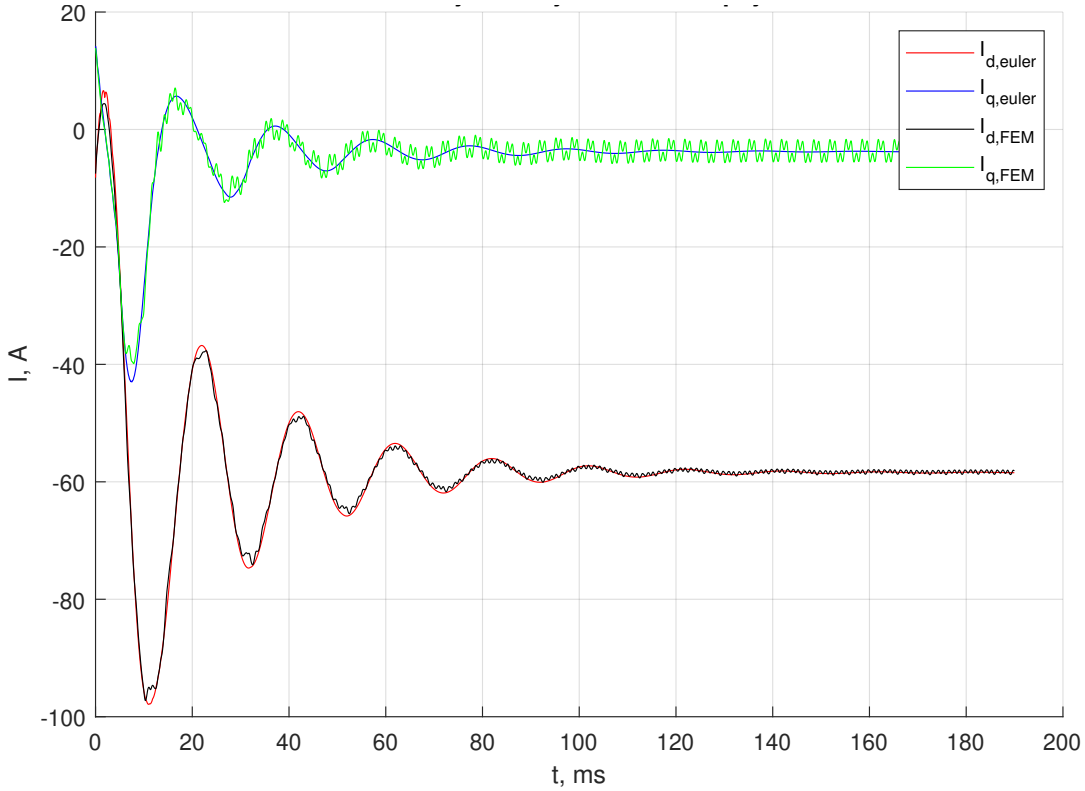
Moment se u modelu određuje analitički, tj. prema jednadžbi 3.4 Točnost izračunatoga momenta potvrdit će se usporedbom s rezultatom FEM simulacije.

$$T_{em} = \frac{3}{2} p \left(\psi_d \cdot i_q - \psi_q \cdot i_d \right) \tag{3.4}$$

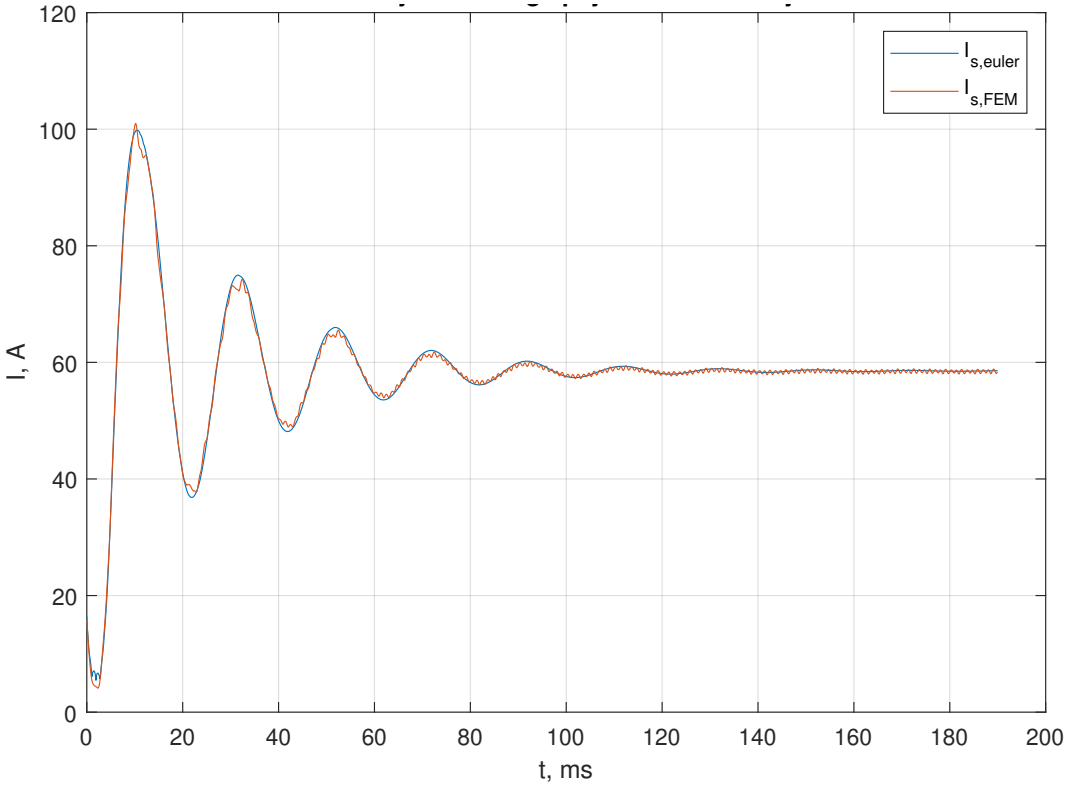
3.4. Usporedba FEM simulacije i Eulerove metode

Eulerova metoda traje par sekundi, a FEM simulacija traje više sati. U ovom poglavlju pokazane su usporedbe valnih oblika struja i ulančenih tokova. Poklapanje tranzijenta opravdava korištenje Eulerove metode, koja je značajno brža.

Na prvoj i drugoj slici prikazane su ovisnosti struja o vremenu simulacije. Najprije su na slici 3.7. uspoređene struje I_d i I_q dobivene FEM simulacijom i Eulerovom metodom. Uspoređena je apsolutna vrijednost struje na slici 3.8. Struje iz FEM simulacije imaju trajnu valovitost u stacionarnom stanju jer se u simulaciji rotacija uzima u obzir.

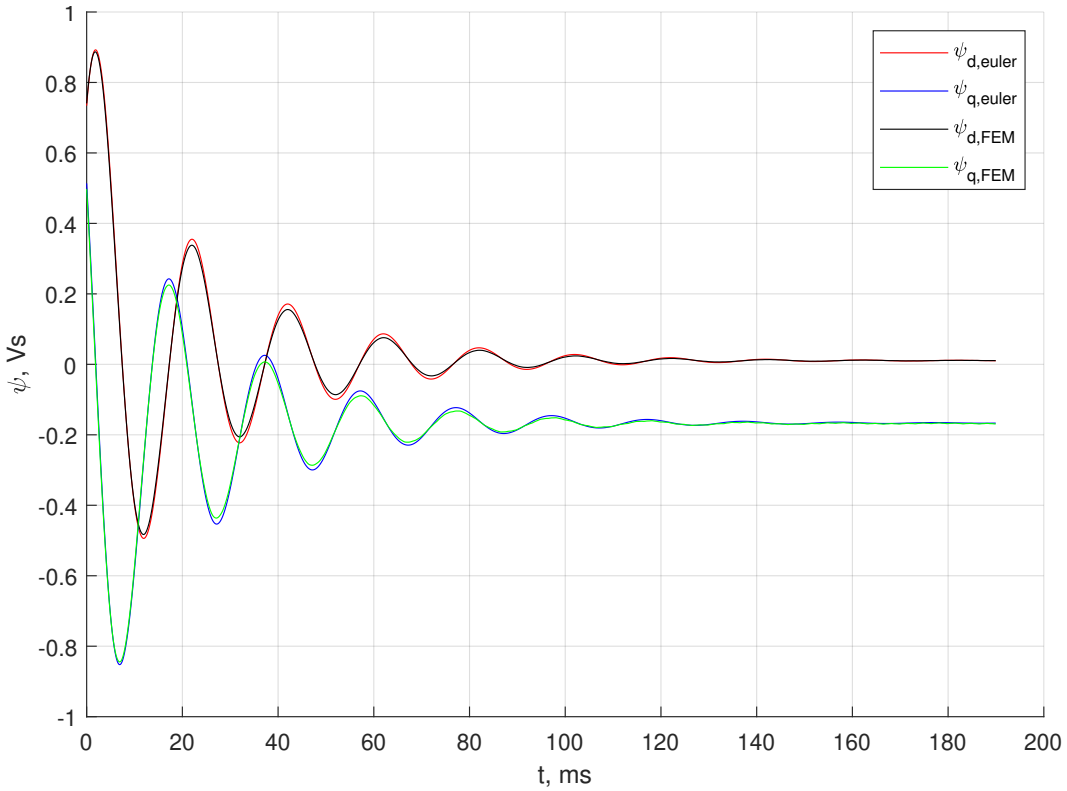


Slika 3.7. Tranzijent struja I_d i I_q u tropolnom kratkom spoju

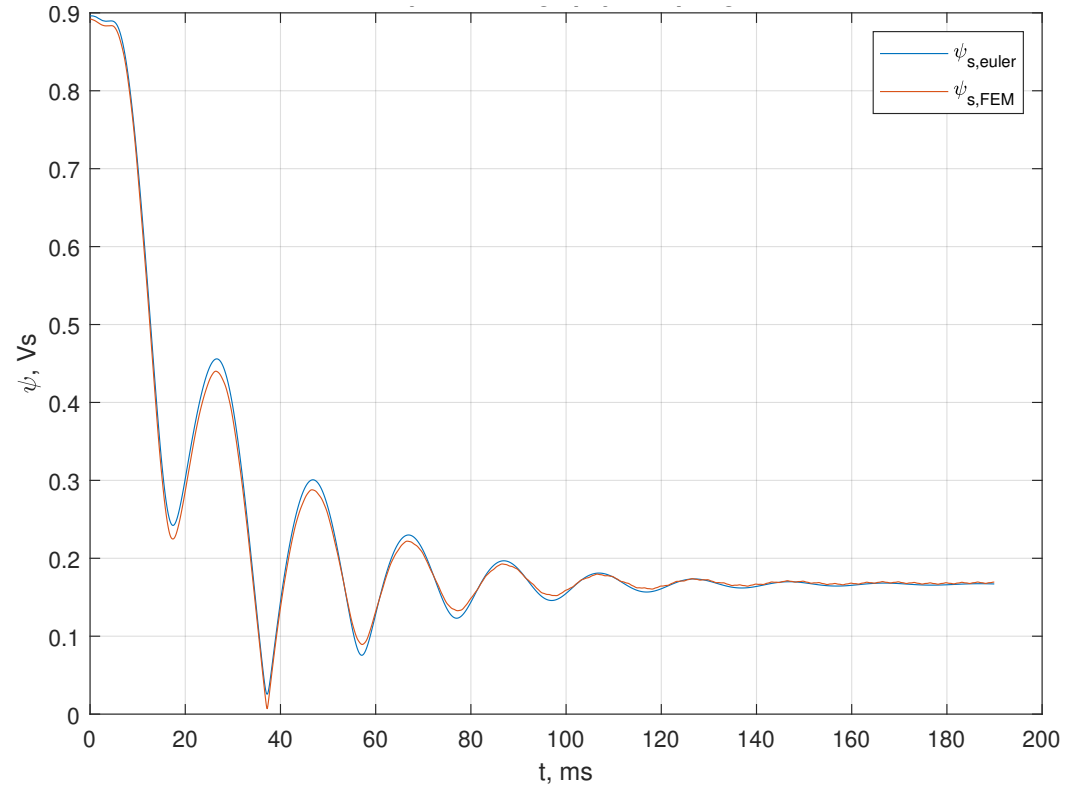


Slika 3.8. Tranzijent struje I_s u tropolnom kratkom spoju

Iduće dvije slike prikazuju usporedbu tranzijenta ulančenih tokova.

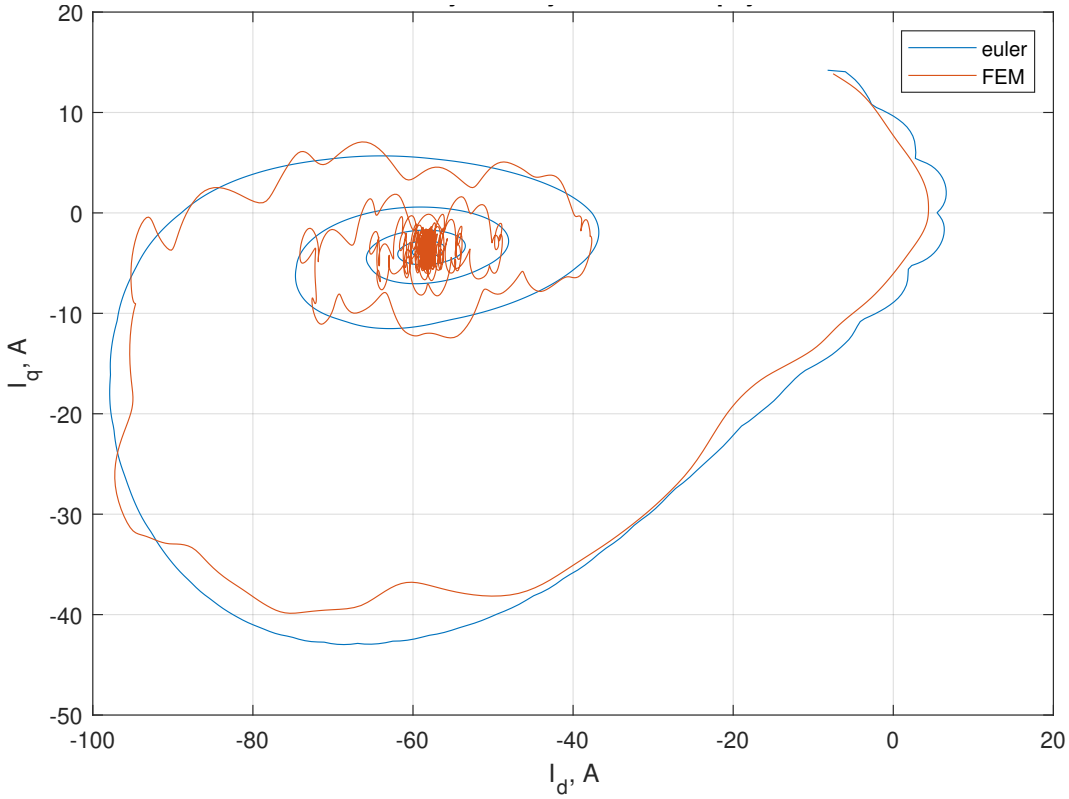


Slika 3.9. Tranzijent ulančenih tokova ψ_d i ψ_q u tropolnom kratkom spoju

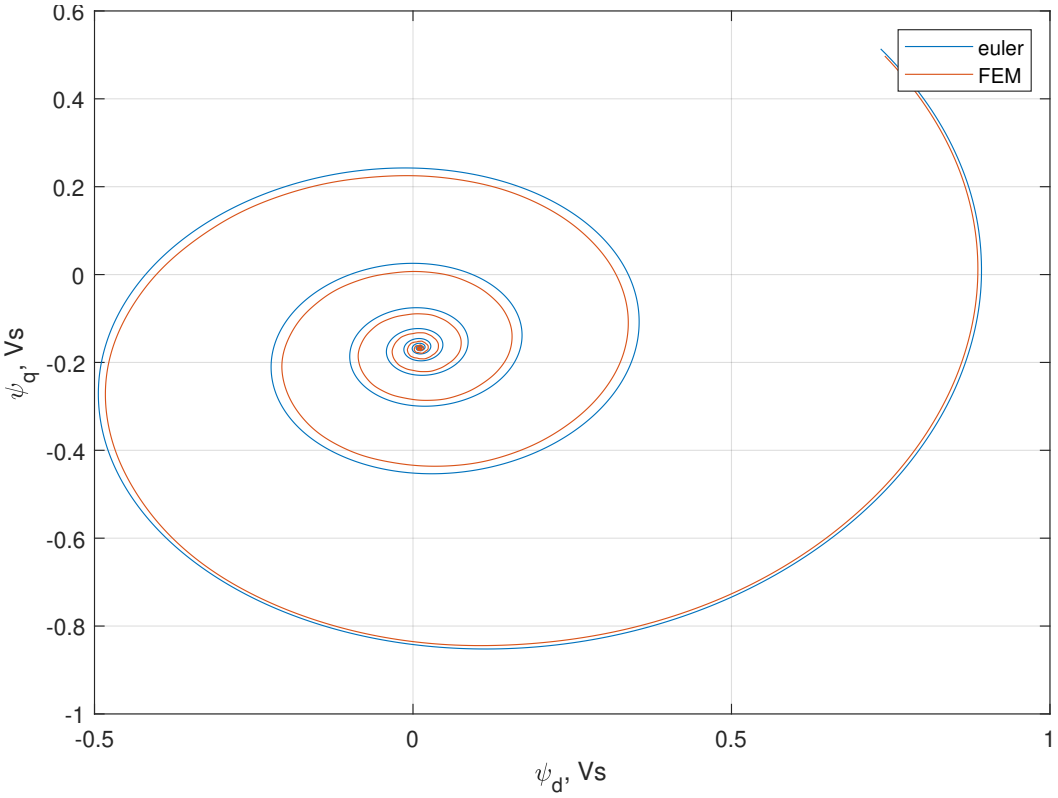


Slika 3.10. Tranzijent ulančenog toka ψ_s u tropolnom kratkom spoju

Valni oblici ulančenih tokova poklapaju se bolje od struja jer nema valovitosti u stacionarnom stanju FEM simulacije. No zbog razlike kod valnih oblika struja, postoji razlika kod ulančenih tokova. Razlika nije ni konstantna ni periodična i može se pripisati numeričkim razlikama FEM simulacije i Eulerove diskretne metode. Dobar prikaz razlike u valnom obliku je tzv. *pužnica*. Struje i ulančeni tokovi se crtaju u kompleksnoj ravnini i tranzijent kratkog spoja ima oblik pužnice koja se savija u točku koja odgovara stacionarnom stanju kratkog spoja. To je prikaz ovisnosti struje I_q o struji I_d, odnosno ulančenog toka ψ_q o toku ψ_d. Slika 3.11. prikazuje pužnicu struje, a slika 3.12. pužnicu ulančenog toka.

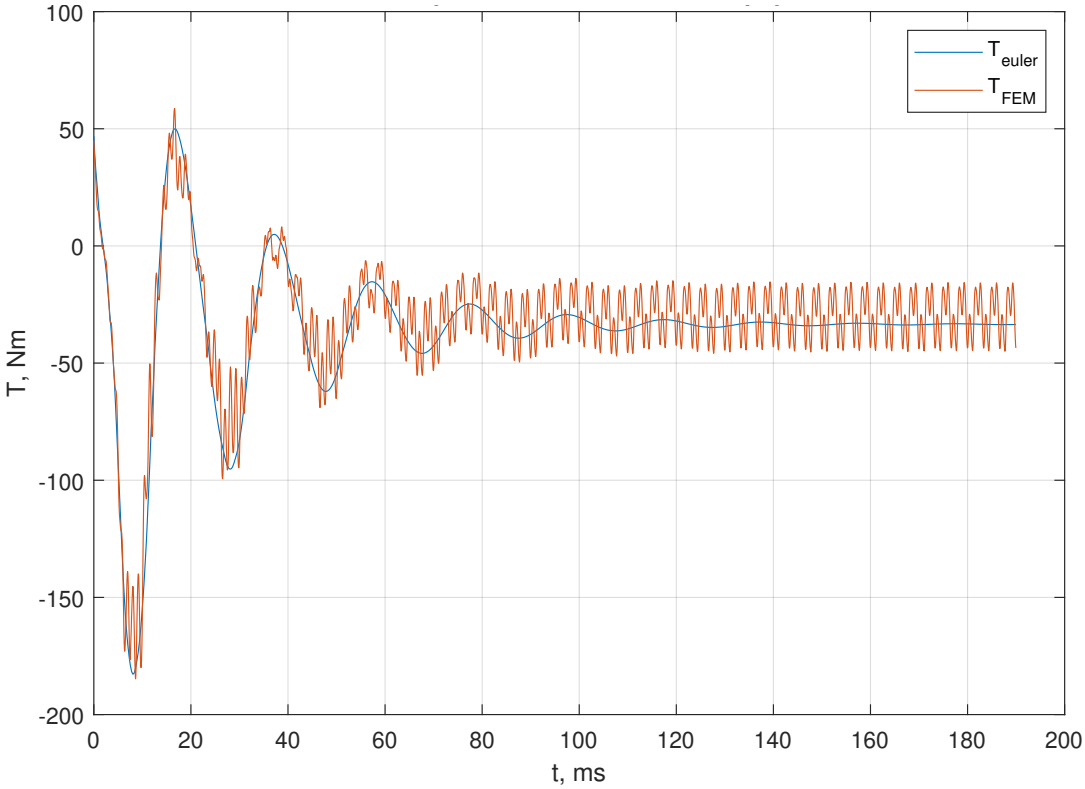


Slika 3.11. Tranzijent struje u tropolnom kratkom spoju, prikazan u kompleksnoj ravnini



Slika 3.12. Tranzijent toka u tropolnom kratkom spoju, prikazan u kompleksnoj ravnini

Zbog valovitosti struje I_q u FEM simulaciji i moment iz FEM simulacije ima valovitost u stacionarnom stanju. Valni oblik momenta prema Eulerovoj metodi prolazi gotovo kroz sredinu oscilacija. Mjesto na kojem valni oblik prema Euleru prolazi kroz oscilacije iz FEM simulacije određeno je položajem rotora pri izradi mape stroja. Ako se mapa radi u nultom položaju rotora, stacionarno stanje prema Euleru ležat će na donjem rubu oscilacija prema FEM simulaciji. Ako je mapa usrednjena prema poziciji rotora, stacionarno stanje prolazit će kroz sredinu oscilacija. Na slici 3.13. prikazan je odnos valnih oblika momenata u slučaju kad je mapa dobivena usrednjavanjem svih položaja nad jednim polom.

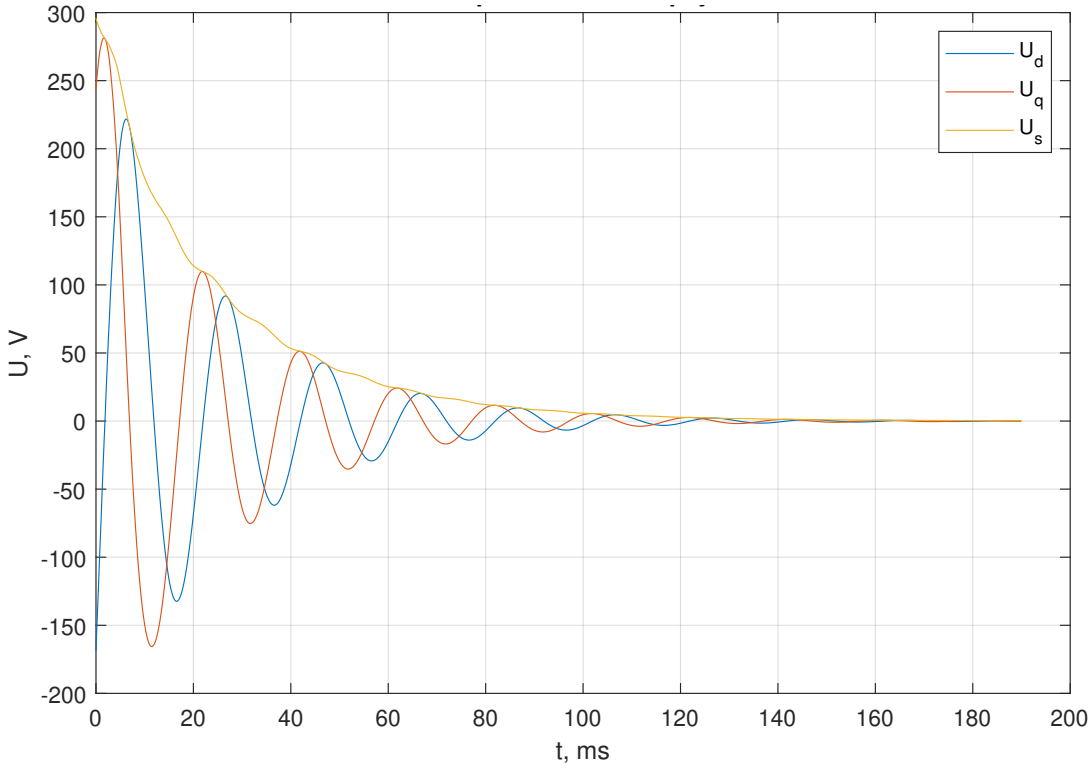


Slika 3.13. Tranzijent momenta u tropolnom kratkom spoju

Dodatno je napravljena validacija Eulerove metode. Provjerava se stacionarno stanje napona u dq sustavu. Prema jednadžbama 3.5 se računaju naponi i za izračun se koriste vektori struja i ulančenih tokova ranije dobiveni Eulerovom metodom.

$$\begin{aligned} u_d &= R_s \cdot i_d - \psi_q \cdot \omega_{el} \\ u_q &= R_s \cdot i_q + \psi_d \cdot \omega_{el} \\ u_s &= \sqrt{u_d^2 + u_q^2} \end{aligned} \tag{3.5}$$

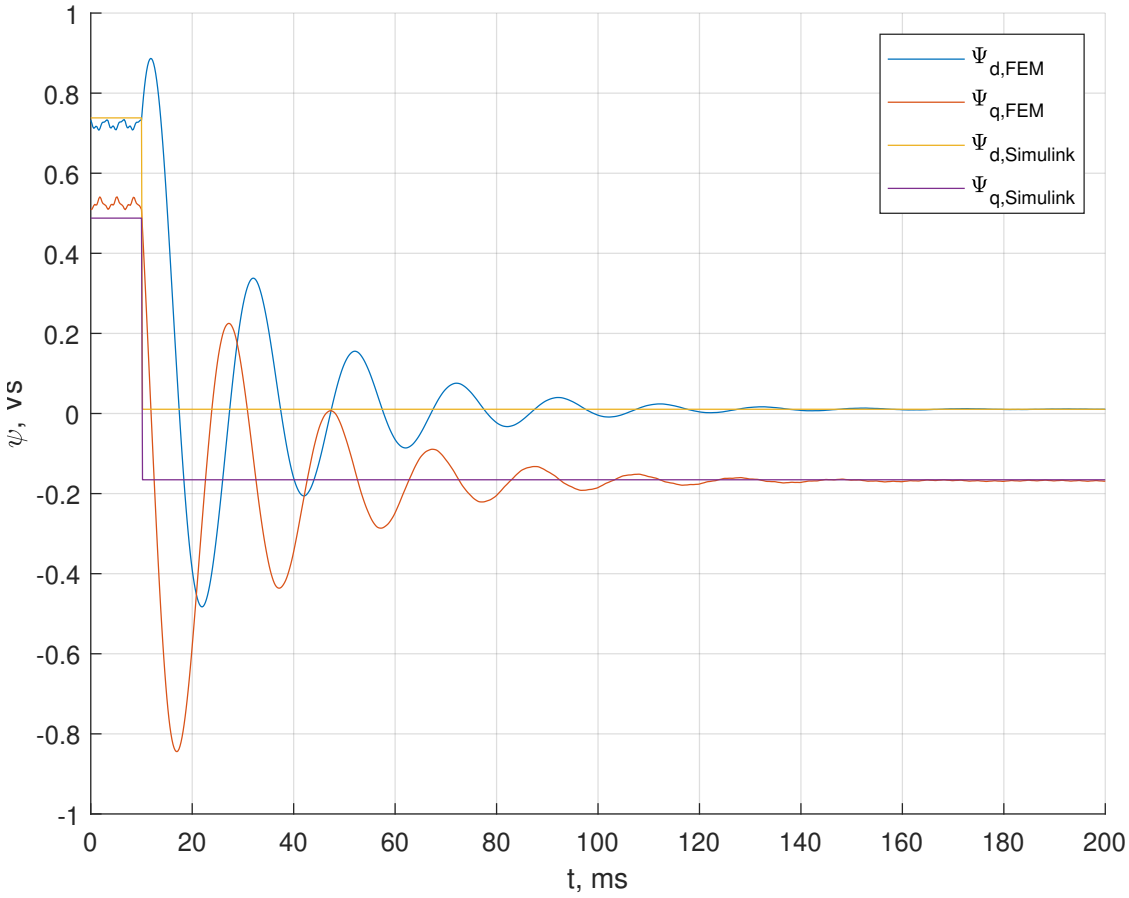
Na jednom grafu 3.14. prikazani su naponi u_d i u_q te apsolutni iznos napona u_s. Prema očekivanju, napon se nakon prijelazne pojave stacionira na nulu.



Slika 3.14. Tranzijent napona na namotu u tropolnom kratkom spoju

3.5. Usporedba Simulink proračuna i FEM simulacije

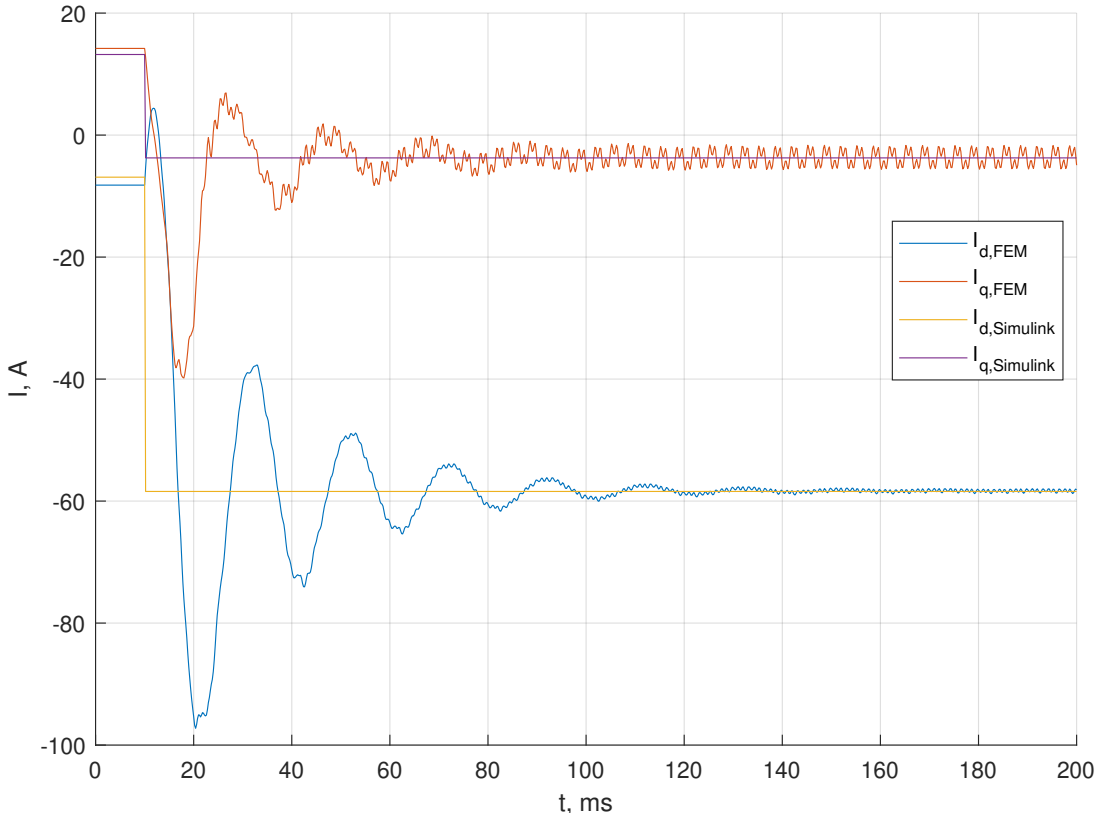
FEM simulacija daje cijeli tranzijent kratkog spoja, a Simulink samo stacionarno stanje. Na slici 3.15. prikazana je usporedba ulančenih tokova u dq sustavu i njihov tranzijent. Stacionarna stanja se točno poklapaju, stoga se preporučuje koristiti Simulink za analizu stacionarne radne točke kratkog spoja. Ako se želi razmatrati tranzijent i demagnetizacija, Simulink neće poslužiti.



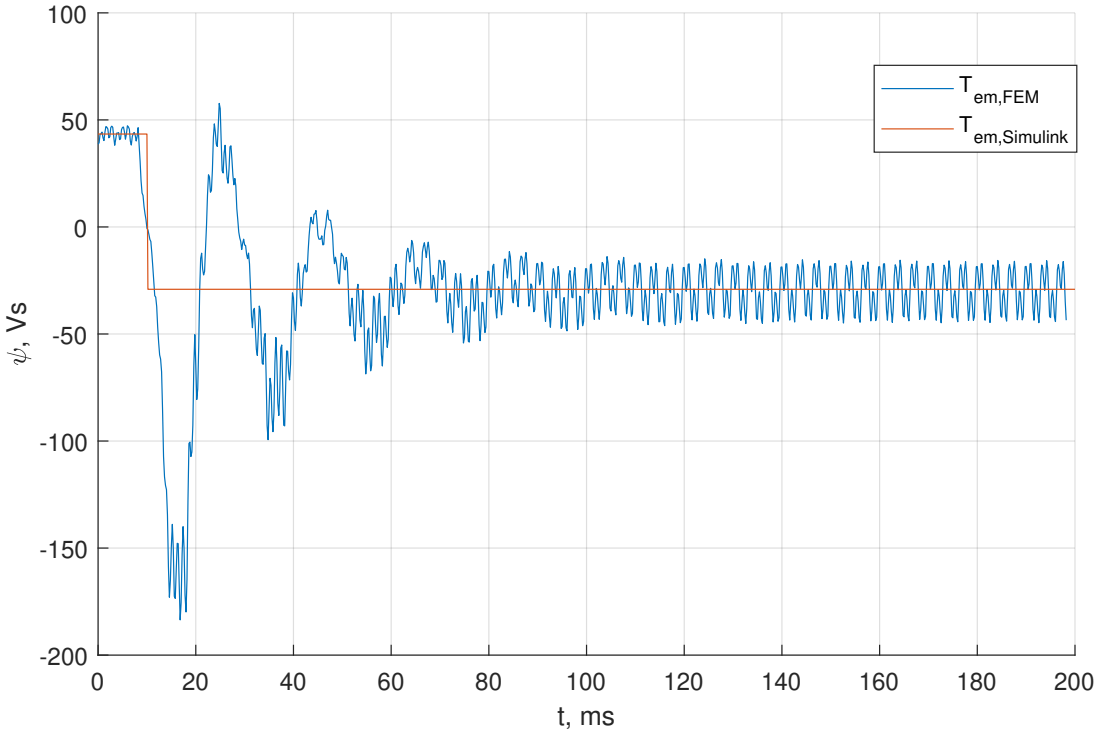
Slika 3.15. Tranzijent ulančenih tokova ψ_d i ψ_q u tropolnom kratkom spoju

Na idućoj slici 3.16. uspoređene su struje u dq sustavu. Struje iz FEM simulacije imaju oscilacije u stacionarnom stanju, a struje iz Simulinka prolaze kroz sredinu oscilacija.

Budući da stacionarna vrijednost struja iz Simulinka prolazi točno kroz sredinu oscilacija FEM simulacije, jednako ponašanje pokazuju valni oblici momenta. Moment je u Simulinku određen računski i na slici 3.17. prikazano je da prolazi točno kroz sredinu oscilacija iz FEM simulacije.



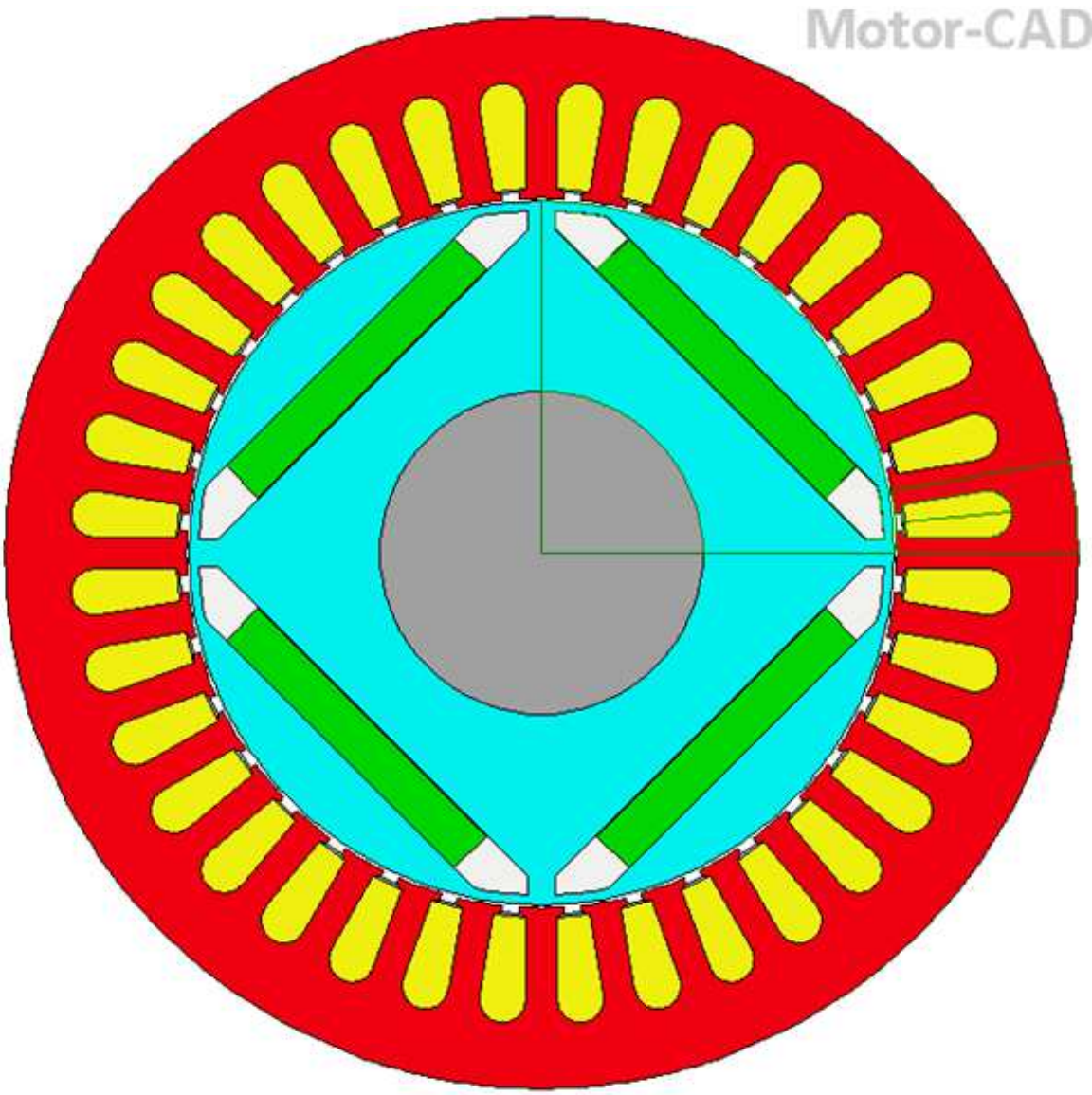
Slika 3.16. Tranzijent struja I_d i I_q u tropolnom kratkom spoju



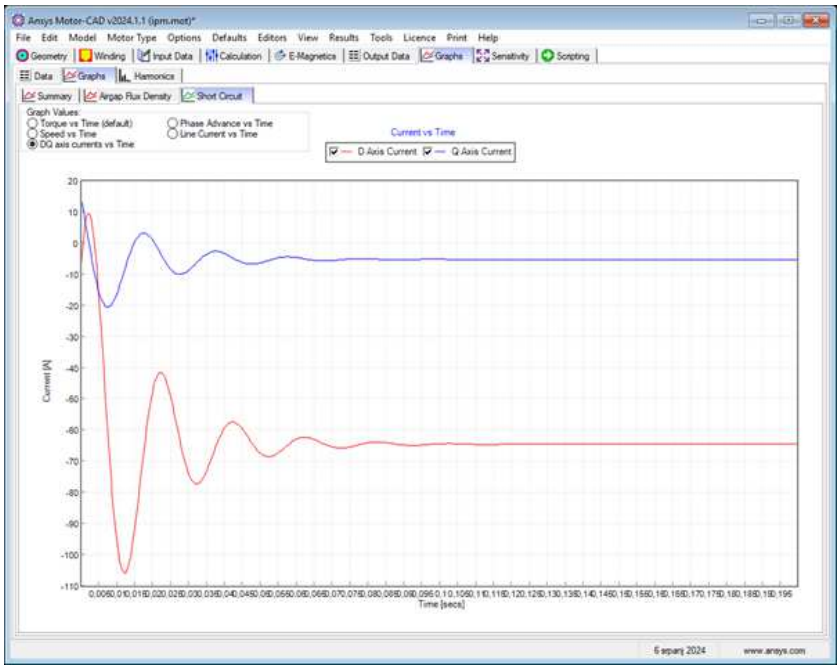
Slika 3.17. Tranzijent momenta u tropolnom kratkom spoju

3.6. Usporedba MotorCAD proračuna i FEM simulacije

MotorCAD je računalni alat kompanije ANSYS koji je napravljen za analizu električnih motora. U njemu je integriran elektromagnetski, mehanički i termički proračun i svi se oni mogu međusobno povezati za vrlo detaljne izračune. Na slici 3.18. prikazan je model IPM motora spreman za analizu. Model je prilagođen tako da bude istovjetan onom iz MAGNET-a. Proračun je zadan na nazivnu radnu točku te je pokrenuta simulacija trenutačnog kratkog spoja. Rezultat analize prikazan je na slici 3.19.

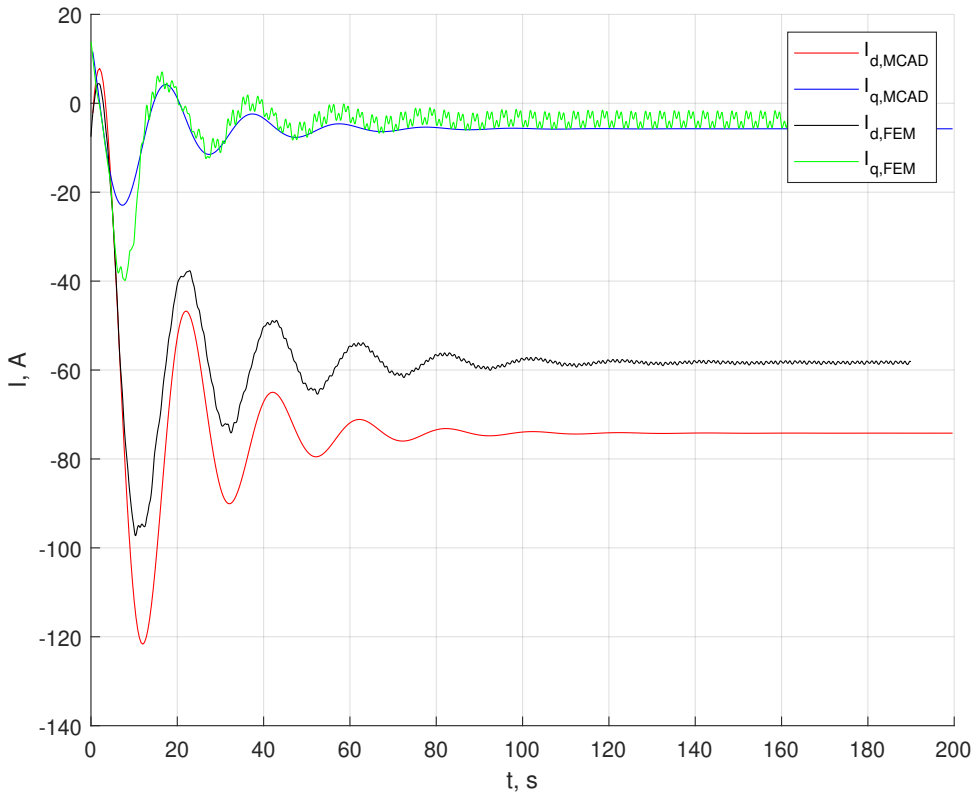


Slika 3.18. Model IPM motora u MotorCAD-u

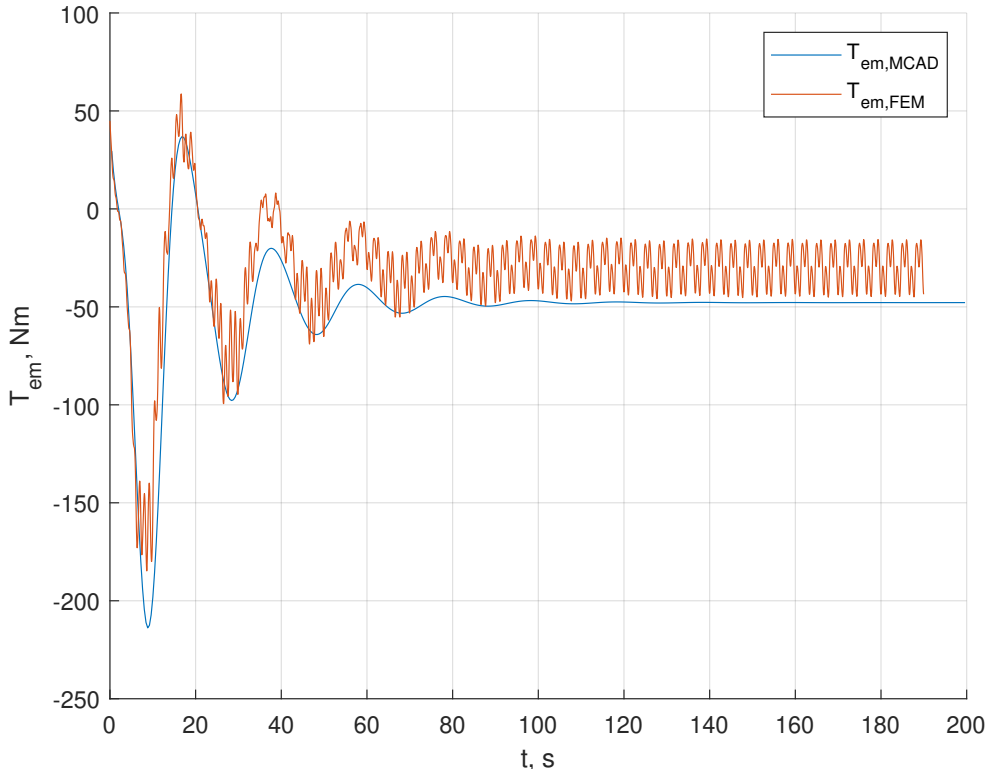


Slika 3.19. Rezultat simulacije kratkog spoja u MotorCAD-u

Većina proračuna u MotorCAD-u temeljena je na FEM analizi, no proračun tranzijenta kratkog spoja nije. Postoji proizvođaču poznat skup interpolacijskih jednadžbi kojima se dolazi do rješenja proračuna kratkog spoja. Iz tog razloga, proračun nije pouzdan i ne slaže se s FEM simulacijom prema MAGNET-u. Na slici 3.20. je uspoređen tranzijent struja kratkog spoja prema MotorCAD-u i MAGNET-u. Na idućoj slici 3.21. uspoređeni su tranzijenti momenata u kratkom spoju. Ostale veličine nisu dostupne u analizi u MotorCAD-u, ali struje i moment dovoljni su pokazatelji neispravnoga rada.

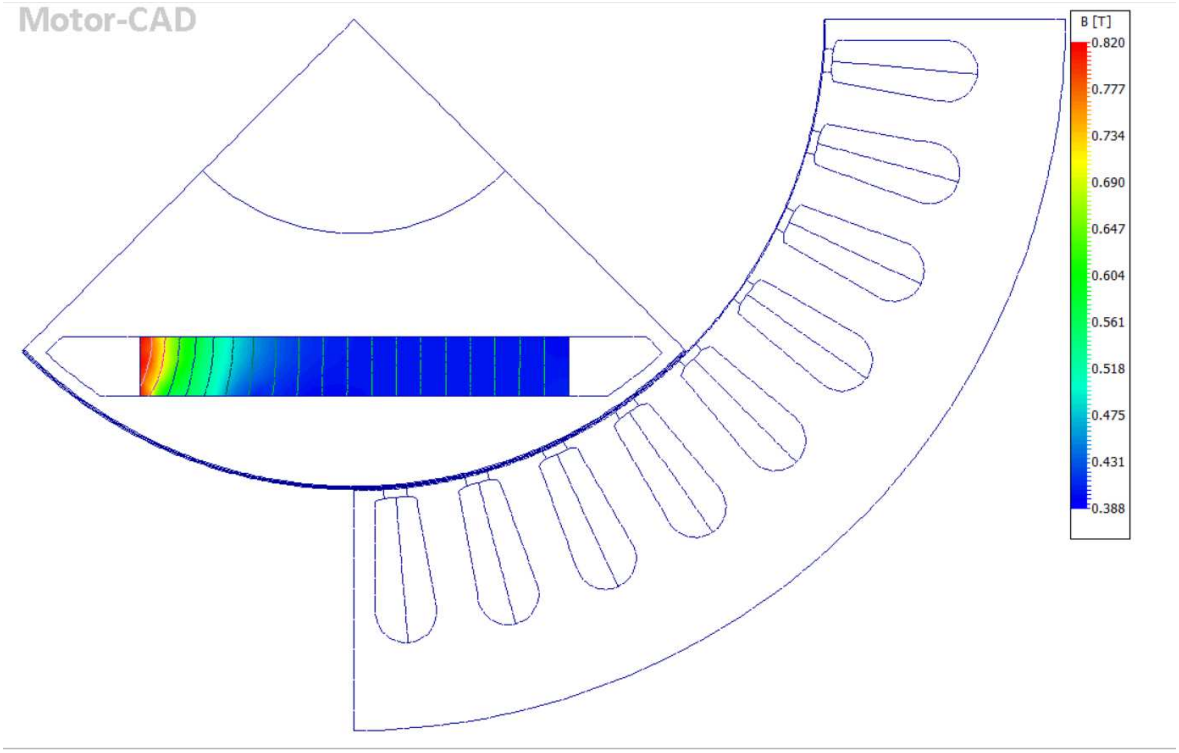


Slika 3.20. Tranzijent struja I_d i I_q u tropolnom kratkom spoju



Slika 3.21. Tranzijent momenta u tropolnom kratkom spoju

MotorCAD ima opciju proračuna demagnetizacije. Bez obzira na to što tranzijent kratkog spoja nije točan, može se provesti analiza demagnetizacije. Zadaje se radna točka koja je dobivena iz FEM simulacije, na kojoj je struja I_d najvećeg iznosa. Slika 3.22. prikazuje gustoću magnetskog toka u magnetu za tu radnu točku. Iznosi B su iznad koljena $B - H$ karakteristike i ne dolazi do demagnetizacije. Na slici 3.23. prikazani su rezultati proračuna. Polje *Magnet demagnetisation ratio* pokazuje koliki postotak magneta je demagnetiziran i u ovom slučaju on je nula.



Slika 3.22. Gustoća magnetskog toka u magnetu u kratkom spoju, najgori trenutak

Losses | Winding | Miscellaneous | Materials

Variable	Value	Units
Br of 1Magnet1N1 at 20 °C	1,31	Tesla
Br of 1Magnet1N1 at 120 °C	1,153	Tesla

Magnet Demagnetization Ratio	0	

Demagnetisation ratio (1Magnet1N1)	0	

Effective Br (1Magnet1N1)	1,153	Tesla

Magnet Flux Density (maximum)	0,3614	Tesla
Magnet Flux Density (average)	0,05894	Tesla
Magnet Flux Density (minimum)	0,004327	Tesla

Magnet Magnetic Field (maximum)	-6,828E005	Amps/m
Magnet Magnetic Field (average)	-8,141E005	Amps/m
Magnet Magnetic Field (minimum)	-8,687E005	Amps/m

Magnet permeance (1Magnet1N1)	0,003963	
Intrinsic permeance (1Magnet1N1)	1,004	

Demagnetisation knee point of 1Magnet1N1	-0,2172	Tesla

Slika 3.23. Rezultat proračuna demagnetizacije u MotorCAD-u

3.7. Programski paket SyR-e

Kao što je ranije spomenuto, tim profesora sa sveučilišta *Politecnico di Torino* posljednjih godina je razvio programski paket za analizu sinkronih električnih strojeva. U toj kategoriji su strojevi s permanentnim magnetima i reluktantni strojevi, po kojima je program dobio ime SyR-e (*eng. Synchronous Reluctance - evolution*). Kod je napisan u Matlabu i sve funkcionalnosti su integrirane u aplikaciju *GUI_Syre_MMM.mlapp*, čija je početna stranica prikazana na slici 3.24. Moguće je učitati jedan od postojećih primjera električnog stroja te provesti sve dostupne proračune radnih točki, izračun gubitaka, tranzijent kratkog spoja i još mnogo drugih opcija. Prilagoditi vlastiti model za analizu sa SyR-e je veliki zadatak jer je potrebno prilagoditi modele za analizu u FEMM-u, izračunati mape stroja i spremiti ih na odgovarajući način u strukture, itd. Iz tog razloga modeli nisu analizirani SyR-e programskim paketom.

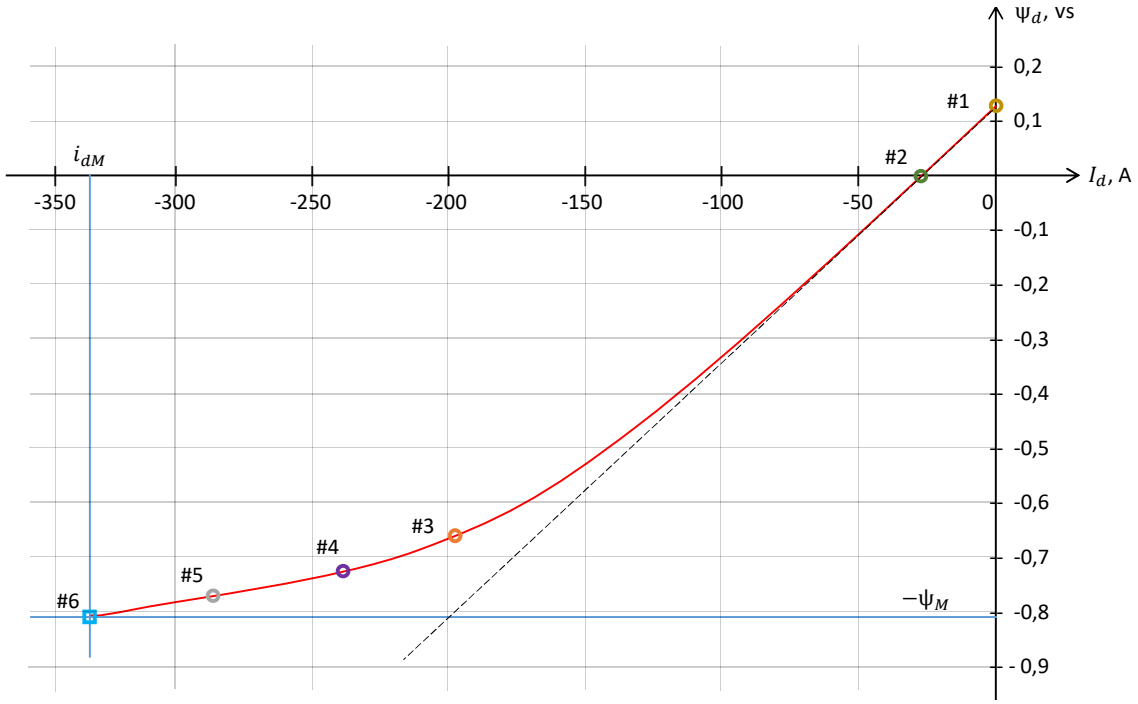
No, kod je detaljno proučen. Cijeli je niz funkcija za obradu mapa stroja, analizu kratkog spoja i provedbu metode sekante. Kod je napisan vrlo općenito i funkcionalnost mu je prilagođena za različite topologije strojeva i razne korisničke zahtjeve. U ovom radu nije korišten ni jedan dio SyR-e koda, samo su analizirane ideje za analizu tranzijenta kratkog spoja i metodu sekante. Preuzeta je ideja za korištenje *scatteredInterpolant* mapa umjesto *griddedInterpolant* i to je bio ključni korak za poklapanje tranzijenta FEM

simulacije i Eulerove metode.

Slika 3.24. Aplikacija programskog paketa SyR-e

4. Metoda sekante

U dosadašnjem proračunu korištene su mape stroja, no sad se želi FEM simulacijom potvrditi hoće li se stroj demagnetizirati u kratkom spoju. Metoda sekante funkcionira iterativno: postupno zadavanje vrijednosti struje I_d i provođenje FEM simulacije kojom se računa ulančeni tok sve dok se ne dosegne iznos negativnog toka magneta $-\psi_M$. Njegova vrijednost određuje se u radnoj točki $(I_d, I_q) = (0, I_s)$ prema $\psi_M = \sqrt{\psi_d^2 + \psi_q^2}$ i predstavlja vrijednost ulančenog toka netom prije demagnetizacije. Slika 4.1. općenito prikazuje ideju metode sekante za određivanje struje I_d koja uzrokuje ulančeni tok iznosa $-\psi_M$ koji je rubna točka demagnetizacije magneta rotora. Svaki tok manjeg iznosa uzrokovat će demagnetizaciju. Crvena krivulja na grafu 4.1. prikazuje krivulju ulančenog toka ψ_d u ovisnosti o struji I_d, prateći svaku točku iteracije.



Slika 4.1. Grafički prikaz metode sekante i pronalaska i_{dM}

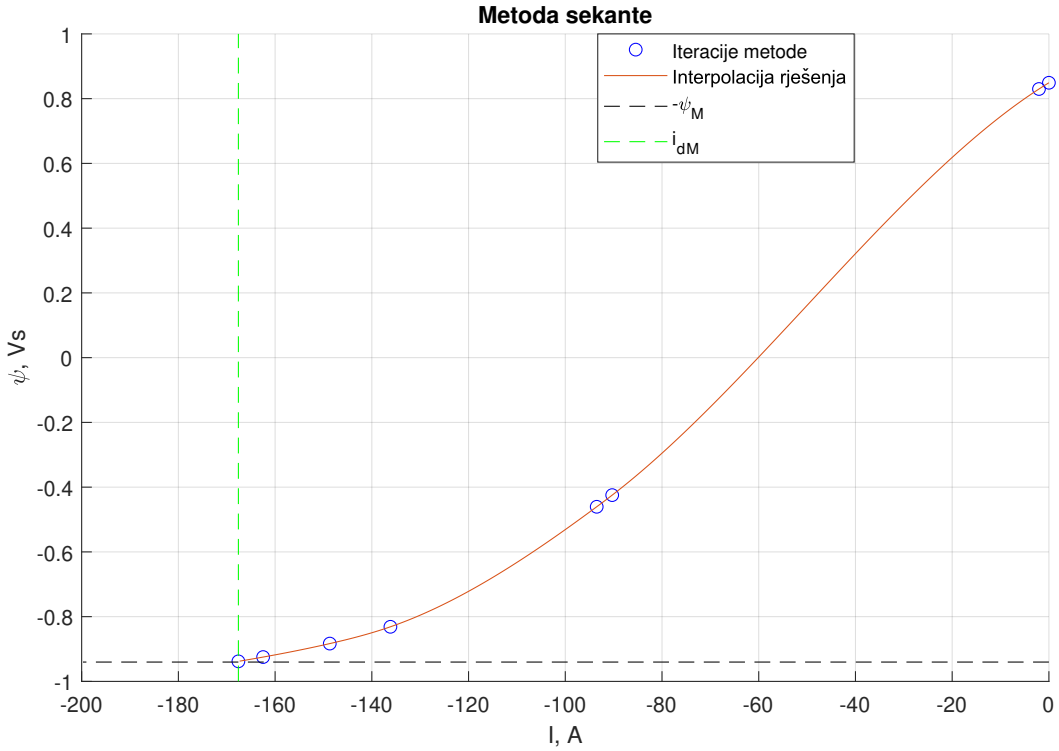
Točke označene na grafu prikazuju FEM simulacije u svakoj iteraciji. Početni koraci metode su točke #1 i #2, one se određuju direktno. Metoda iterativno traži presjek pravca kroz dvije točke s pravcem $-\psi_M$, što je ilustrativno prikazano crtkanim pravcem na grafu 4.1., koji je opisan jednadžbom 4.1 Ako se traži presjecište, onda se $\psi_{d,i+2}$ izjednači s $-\psi_M$ i rješenje jednadžbe je i_{i+2}.

$$\psi_{d,i+2} - \psi_{d,i+1} = \frac{\psi_{d,i} - \psi_{d,i+1}}{i_{d,i} - i_{d,i+1}} \left(i_{d,i+2} - i_{d,i+1}\right) \tag{4.1}$$

Struja i_{i+2} koja odgovara presjecištu ulazi u novu FEM simulaciju i računa se odgovarajući ulančeni tok ψ_d, koji čini novu točku kroz koju se crta novi pravac i traži novo presjecište. Postupak se ponavlja sve dok se ne nađe struja i_{dM} za koju je ulančeni tok unutar 1% pogreške jednak $-\psi_M$.

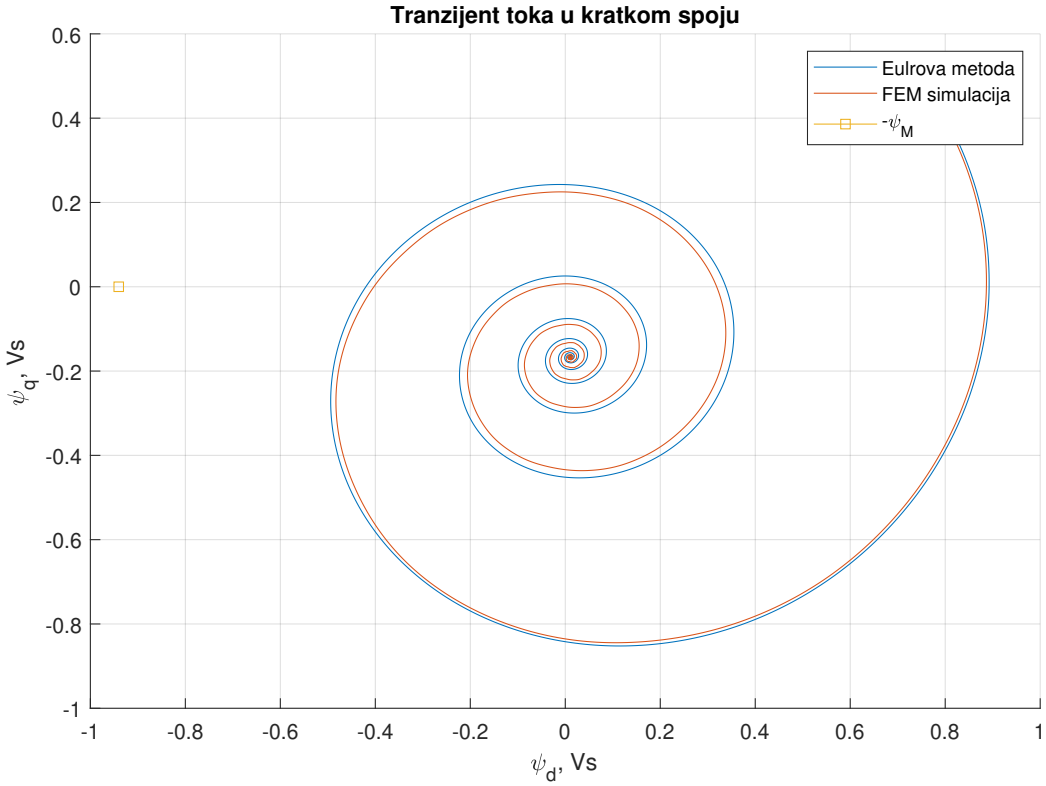
U ovom radu su za početne dvije točke odabrane radne točke $(0, 0)$ i $\left(\frac{I_{d,naz}}{4}, 0\right)$. Moguće je odabrati bilo koji drugi uređeni par, ali bitno je da struja I_q bude nula, da se sve odvija u d-osi. Uz to, metoda je prilagođena tako se postepeno približava $-\psi_M$. Time se osigurava postepeno povećavanje struje i sprječava se preskakanje konačne vrijednosti. Prve dvije FEM simulacije određuju početne uvjete prema kojima se crta pravac. Presjecište se traži s pravcem $\psi_d = 0\,\mathrm{Vs}$ i dobiva se struja za treću FEM simulaciju. U idućem koraku, nakon treće FEM simulacije, traži se presjecište pravca koji prolazi kroz drugu i treću točku te pravca $\psi_d = -\frac{\psi_M}{2}$ što daje struju četvrte točke. Nakon četvrte FEM simulacije definiran je četvrti uređeni par struje i ulančenog toka. U svakom idućem koraku traži se presjecište pravca kroz prethodna dva uređena para te pravca $\psi_d = -\psi_M$.

Slika 4.2. prikazuje rezultate metode sekante na primjeru IPM stroja. U šest iteracija pronađena je struja i_{dM}, sveukupno osam točaka je potrebno za metodu sekante. Uključujući i FEM simulaciju za određivanje ψ_M, odrađeno je devet FEM simulacija, ukupnog trajanja 2,5 min. Uređeni par rješenja je $(\psi_M, i_{dM}) = (-0{,}94\,\mathrm{Vs}, -166{,}92\,\mathrm{A})$.



Slika 4.2. Rezultati metode sekante za IPM motor

Da u kratkom spoju IPM motora neće doći do demagnetizacije, grafički prikazuje slika 4.3. Ponovno se prikazuje tranzijent ulančenog toka u kratkom spoju u kompleksnoj ravnini. Na ovom prikazu dodana je točka $(-\psi_M, 0)$ koja je prikazana žutim kvadratićem. Točka je dovoljno daleko od tranzijenta, što pokazuje sigurnost da neće doći do demagnetizacije.

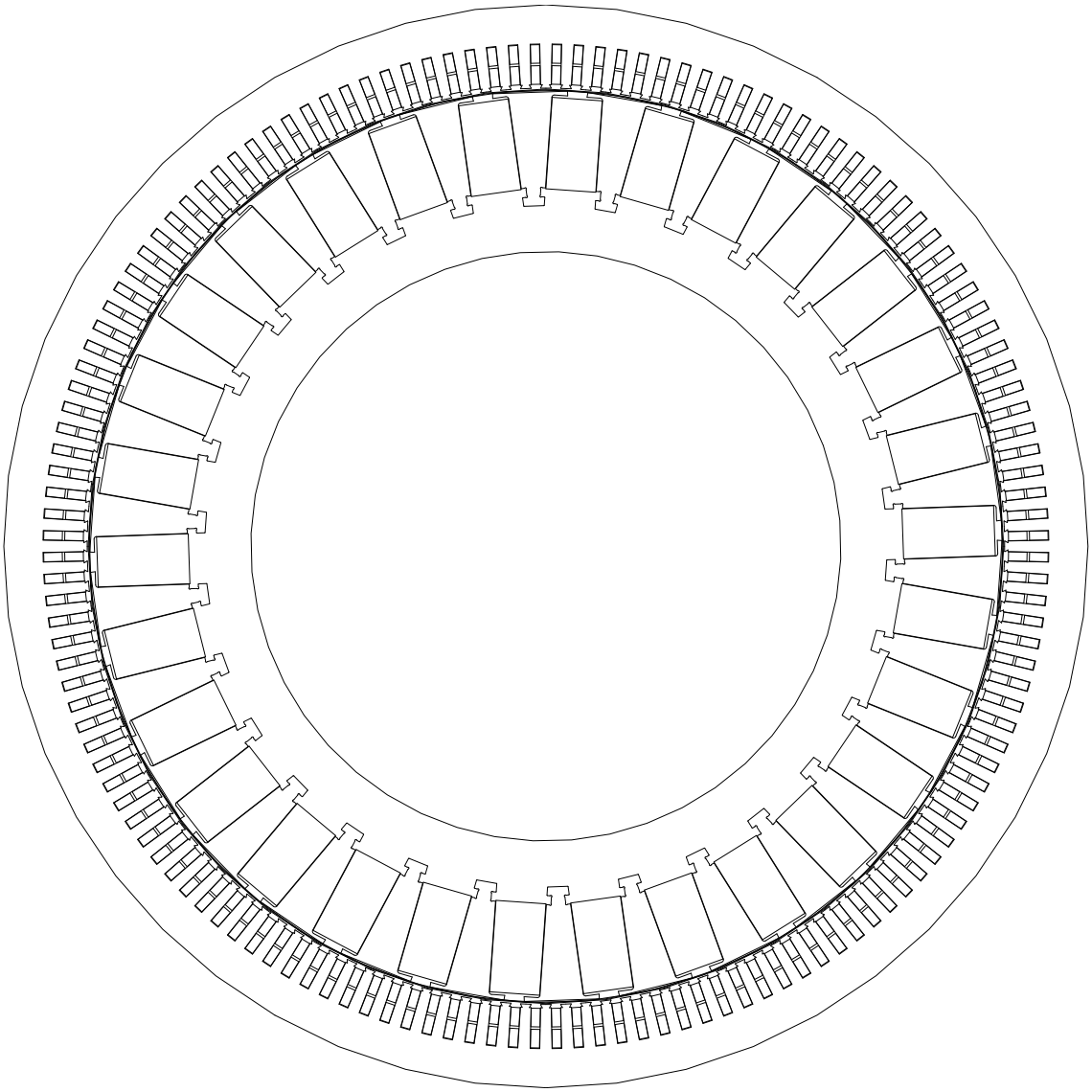


Slika 4.3. Usporedba rubne točke demagnetizacije $-\psi_M$ s tranzijentom ulančenog toka u kratkom spoju

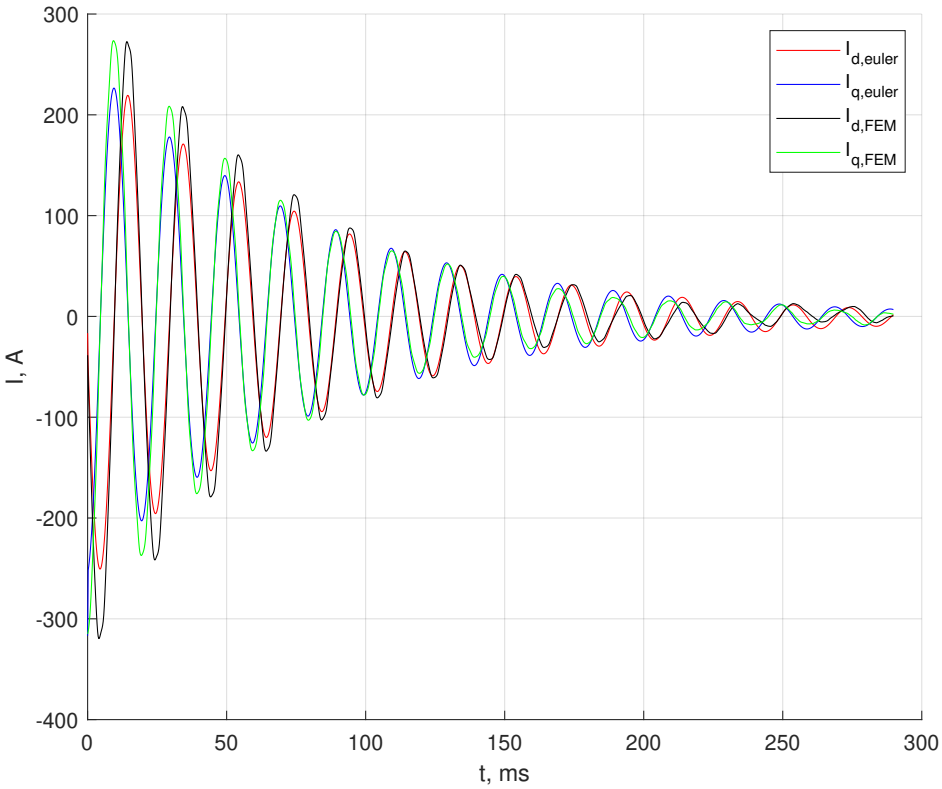
5. Proračun tranzijenta kratkog spoja generatora

U ovom su poglavlju prikazani rezultati proračuna tranzijenta kratkog spoja za drugačiju topologiju stroja. Generator je sinkroni stroj s unutrašnjim permanentnim magnetima prikazan na slici 5.1. Stroj ima 30 polova, 144 utora nazivna struja iznosi 224 A_{rms}, otpor faze je 0,0811 Ω i namot je spojen serijski. Nazivni napon generatora je 380 V na frekvenciji 50 Hz. Nazivna radna točka je za $\gamma = -177°$.

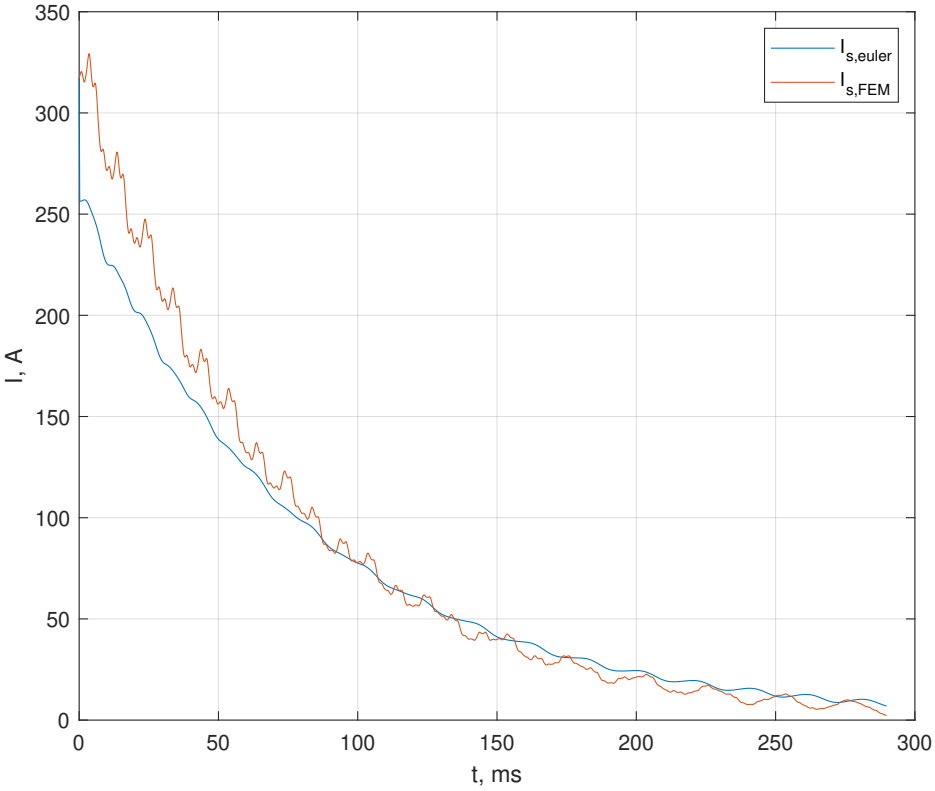
U glavnu skriptu su upisani traženi podaci. Modeli su pripremljeni kako je opisano u poglavlju 2.3. te je napravljen proračun tranzijenta kratkog spoja. U nastavku poglavlja prikazani su rezultati usporedbe FEM simulacije kratkog spoja i Eulerove analitičke metode. Na slikama 5.2. i 5.3. prikazane su tranzijentne struje kratkog spoja. Tranzijent ulančenih tokova prikazan je na slikama 5.4. i 5.5. U kompleksnoj ravnini su prikazane struje na slici 5.6. i ulančeni tokovi na slici 5.7. Na posljednjoj slici 5.8. prikazan je moment u kratkom spoju.



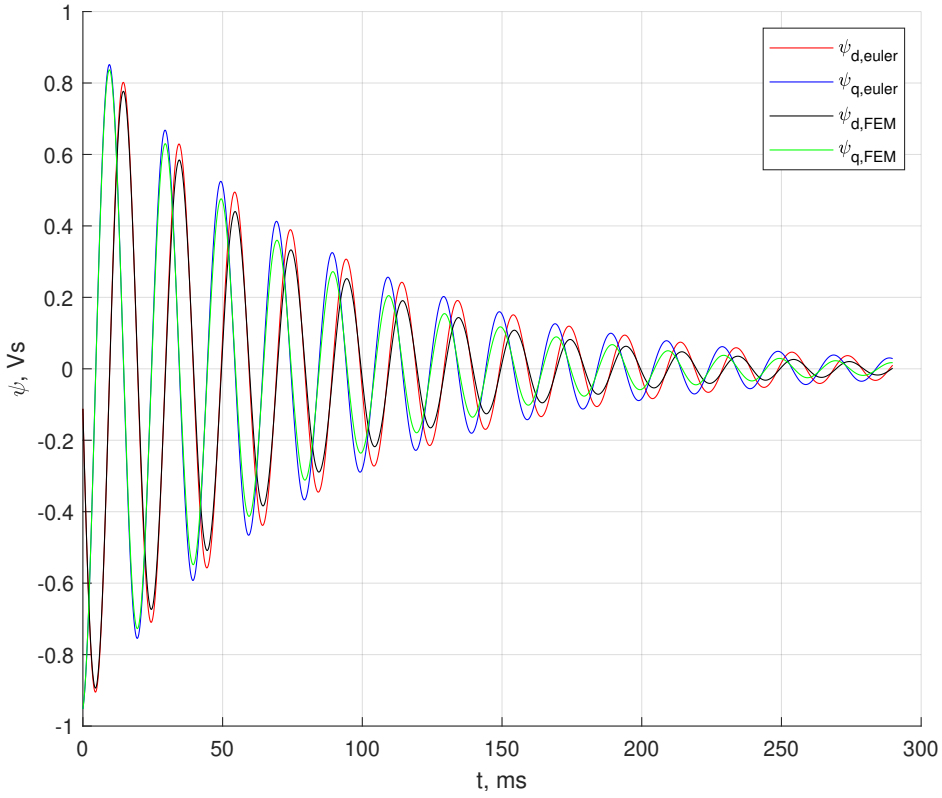
Slika 5.1. Poprečni presjek IPM generatora



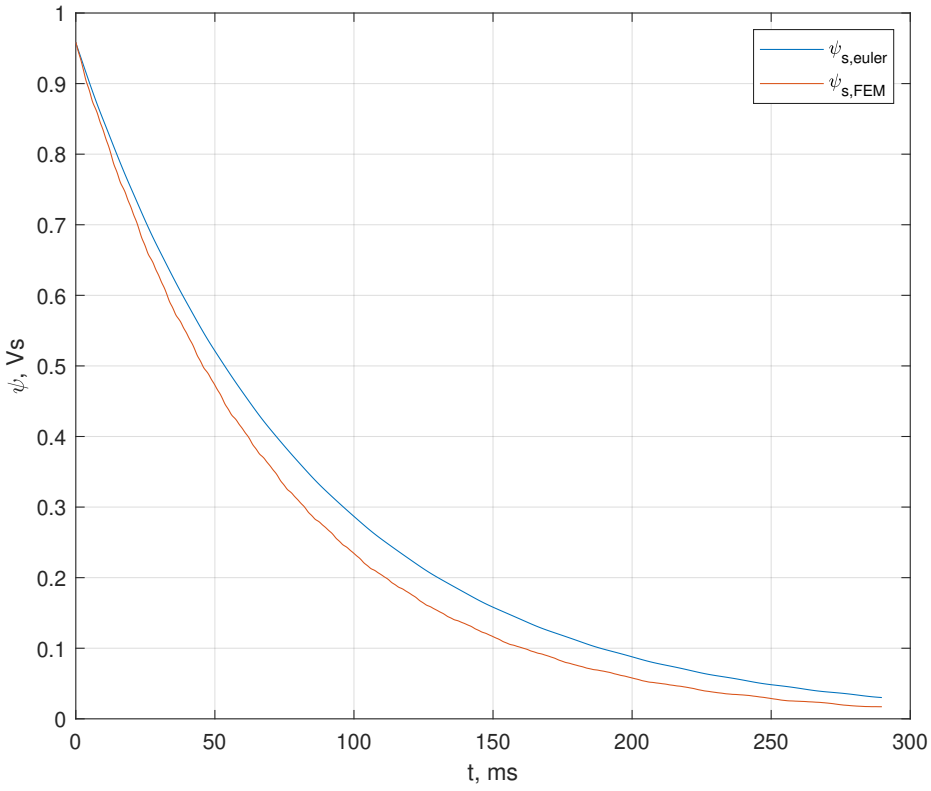
Slika 5.2. Tranzijent struja I_d i I_q u tropolnom kratkom spoju



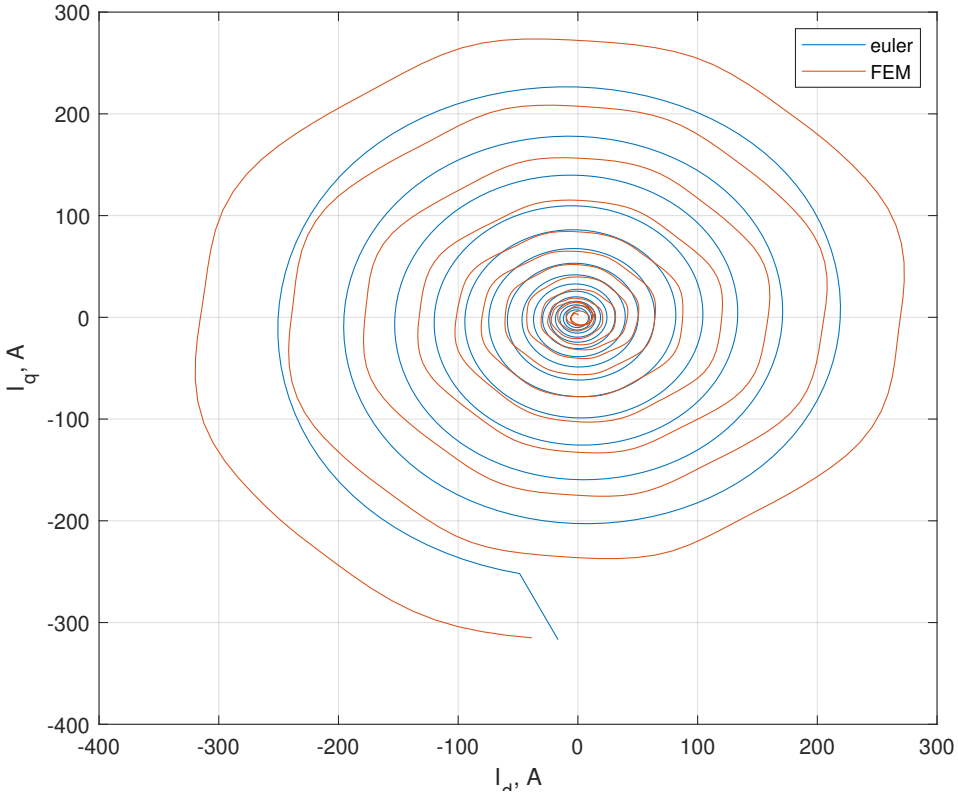
Slika 5.3. Tranzijent struje I_s u tropolnom kratkom spoju



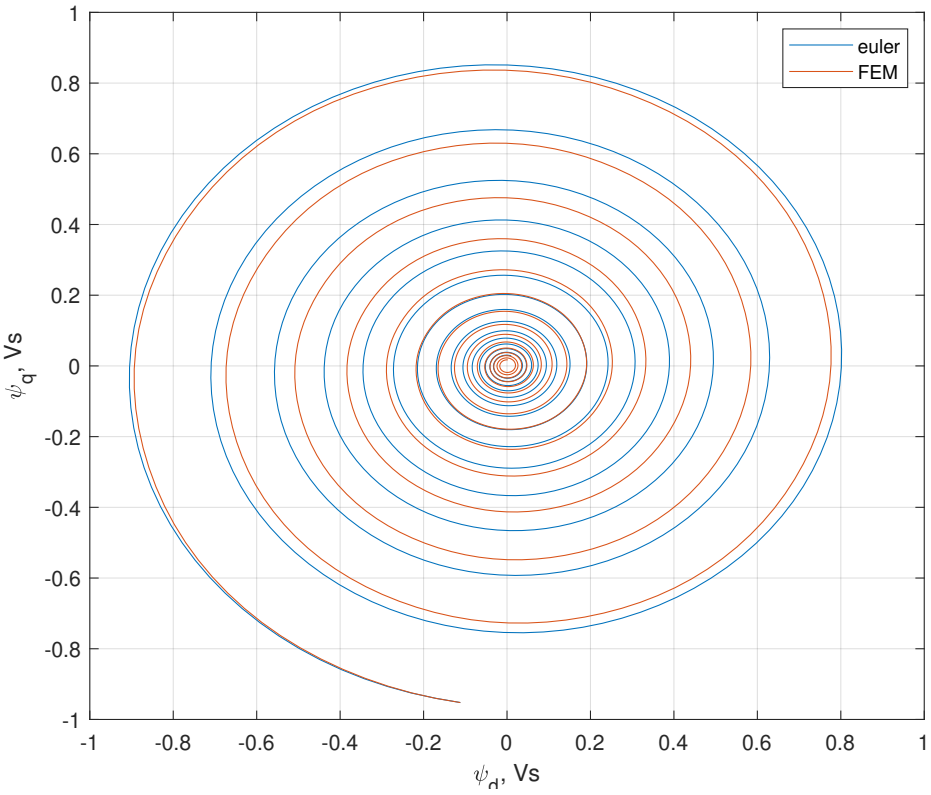
Slika 5.4. Tranzijent ulančenih tokova ψ_d i ψ_q u tropolnom kratkom spoju



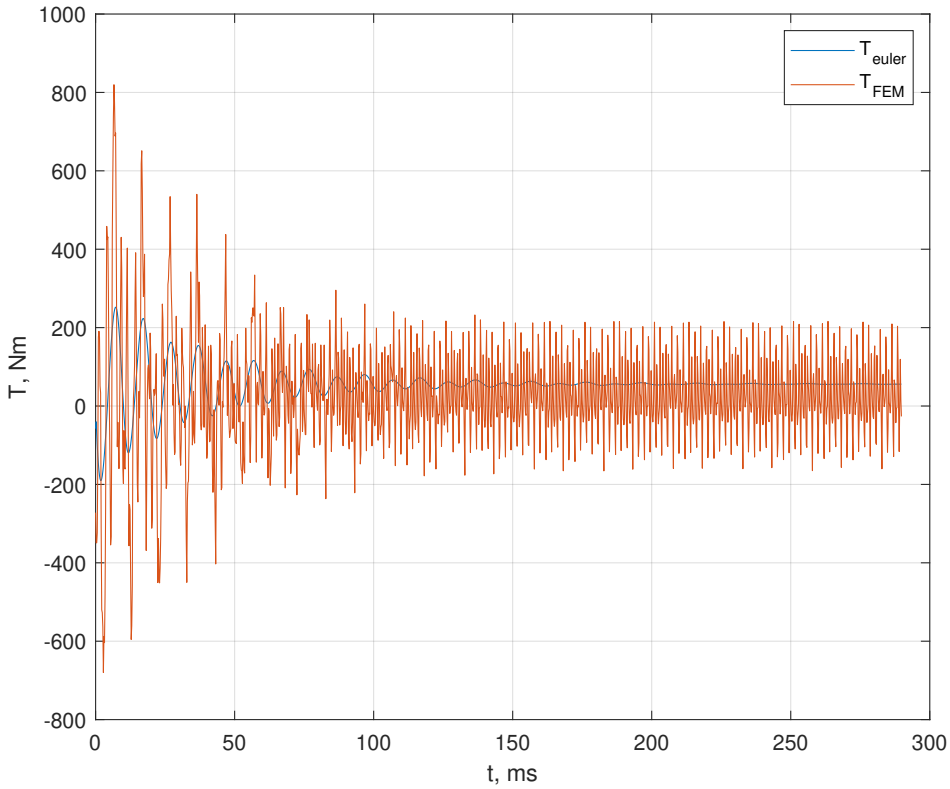
Slika 5.5. Tranzijent ulančenog toka ψ_s u tropolnom kratkom spoju



Slika 5.6. Tranzijent struje u tropolnom kratkom spoju, prikazan u kompleksnoj ravnini

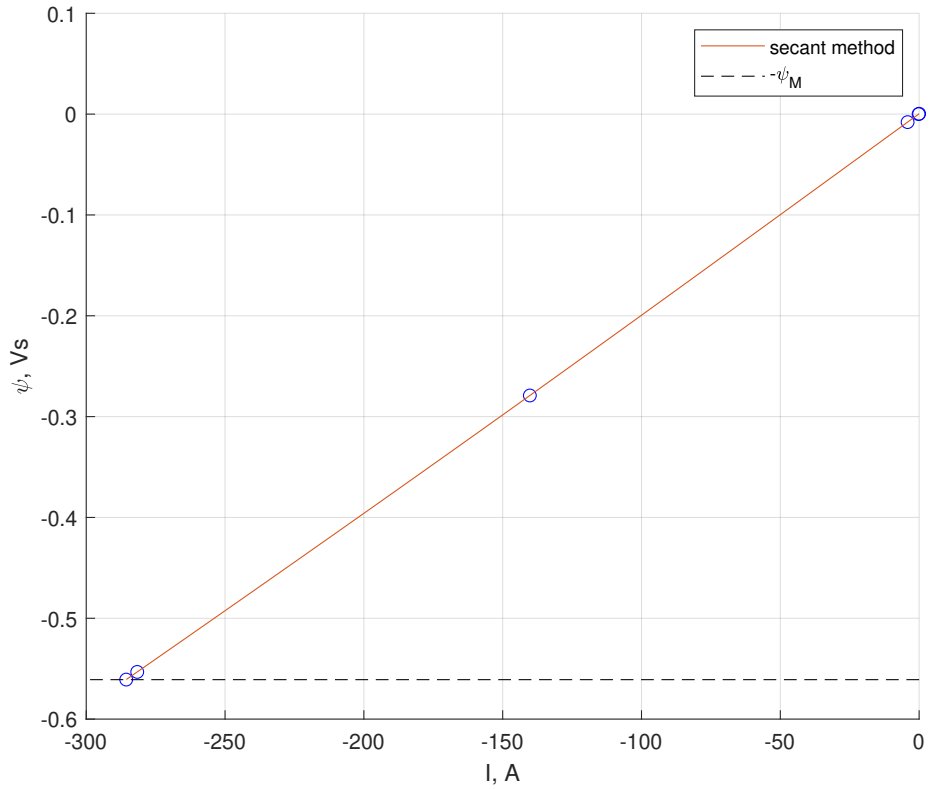


Slika 5.7. Tranzijent toka u tropolnom kratkom spoju, prikazan u kompleksnoj ravnini

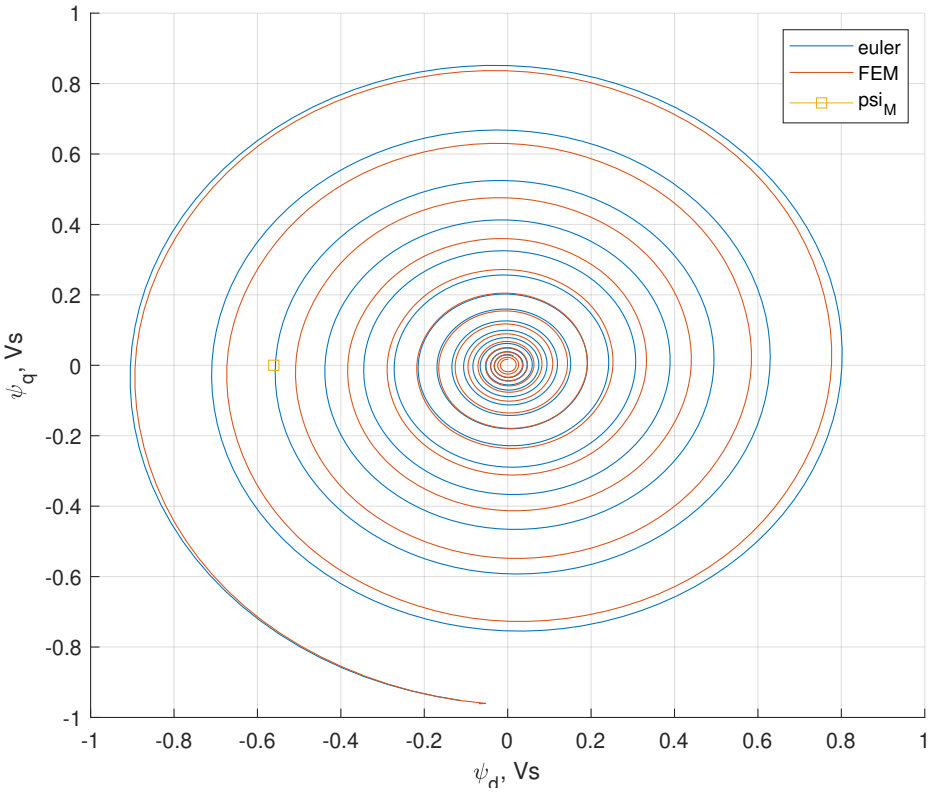


Slika 5.8. Tranzijent momenta u tropolnom kratkom spoju

Napravljena je metoda sekante te su rezultati prikazani na slikama 5.9. i 5.10. U slučaju kratkog spoja, magneti na rotoru će se demagnetizirati.



Slika 5.9. Rezultati metode sekante za IPM generator



Slika 5.10. Usporedba rubne točke demagnetizacije $-\psi_M$ s tranzijentom ulančenog toka u kratkom spoju

6. Zaključak

Strojevi s permanentnim magnetima zahtijevaju dodatne analize u slučaju kratkog spoja. Magneti na rotoru mogu biti oštećeni, tj. demagnetizirani zbog utjecaja struja kratkog spoja. U ovom su radu analizirani različiti načini određivanja tranzijenta struja kratkog spoja te njihov utjecaj na magnete.

FEM simulacija najpouzdaniji je način za određivanje tranzijenta kratkog spoja, no vrlo dugotrajan. Za kvalitetnu analizu potrebno je nekoliko sati te su potrebna brža i jednako dobra rješenja. Eulerova analitička metoda temeljena na mapama ulančanih tokova pokazuje se vrlo učinkovita i vrlo brza. Izrada mapa je dugotrajna, ali mape imaju mnoge načine upotrebe. Postoji opcija korištenja 3D mapa koje zahtijevaju najviše vremena za izradu, ali rezultati su precizniji. Druga opcija je korištenje 2D mapa gdje su efekti pomaka rotora zanemareni, ali je vrijeme izrade značajno kraće. Ako su mape spremne, Eulerovom metodom tranzijent se odredi za svega nekoliko sekundi. Simulink je jednako brzo rješenje, no njime se pokazuje samo stacionarno stanje, a ne cijeli tranzijent. Iz tog je razloga njegova upotreba ograničena. MotorCAD ima brzi softver za određivanje struja kratkog spoja, no netočan je. Stoga se preporučuje proračun tranzijenta temeljen na mapama.

Demagnetizacija magneta određuje se prema koljenu B-H karakteristike koje ovisi o temperaturi magneta. No, za analizu radnih točaka na rubu demagnetizacije, može se promatrati ulančeni tok ψ_d koji je jednak negativnom iznosu toka magneta. Metodom sekante može se odrediti struja I_d za koju se postiže taj ulančeni tok. U nekoliko iteracija FEM analize dolazi se do rješenja. Najveća prednost metode sekante je ta da nije potrebno poznavanje mapa stroja, već je dovoljan samo model stroja.

Literatura

[1] G. Choi i T. M. Jahns, "Demagnetization characteristics of permanent magnet synchronous machines", u *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, 2014., str. 469–475.

[2] A. Naina, S. Paryani, i S. S. N. Jani, "Comparison between surface-mounted and interior pm motor for ev application", u *2021 International Conference on Intelligent Technologies (CONIT)*, 2021., str. 1–6.

[3] Y. Shi, "Fault analysis of permanent magnet synchronous machines for safety critical applications", doktorska disertacija, The University of Sheffield, 2019.

[4] G. Zhao, L. Tian, Q. Shen, i R. Tang, "Demagnetization analysis of permanent magnet synchronous machines under short circuit fault", u *2010 Asia-Pacific Power and Energy Engineering Conference*, 2010., str. 1–4.

[5] S. Sakunthala, R. Kiranmayi, i P. N. Mandadi, "A study on industrial motor drives: Comparison and applications of pmsm and bldc motor drives", u *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017., str. 537–540.

[6] S. Ferrari, G. Dilevrano, P. Ragazzo, P. Pescetto, i G. Pellegrino, "Fast determination of transient short-circuit current of pm synchronous machines via magnetostatic flux maps", *IEEE Transactions on Industry Applications*, sv. 59, br. 4, str. 4000–4009, 2023.

[7] *'Simcenter MAGNET - Introduction to Magnetostatic 2D Modeling'*.

[8] S. Ferrari, G. Dilevrano, P. Ragazzo, i G. Pellegrino, “The dq-theta flux map model of synchronous machines”, u *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2021., str. 3716–3723.

[9] S. Ferrari, P. Ragazzo, G. Dilevrano, i G. Pellegrino, “Flux-map based fea evaluation of synchronous machine efficiency maps”, u *2021 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD)*, 2021., str. 76–81.

[10] S. Ferrari, G. Dilevrano, P. Ragazzo, i G. Pellegrino, “Determination of the symmetric short-circuit currents of synchronous permanent magnet machines using magnetostatic flux maps”, u *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2021., str. 3697–3704.

[11] prof. dr. sc. Damir Žarko i prof. dr. sc. Stjepan Stipetić, “Metoda konačnih elemenata: teorija i praktične primjene u elektrostrojarstvu”, travanj 2024.

Sažetak

Proračun tranzijentne struje kratkog spoja sinkronog stroja s permanentnim magnetima korištenjem mapa ulančenih tokova

Kristijan Gaži

U diplomskom radu analiziran je tranzijent struja i ulančenih tokova tijekom tropolnog kratkog spoja. Tranzijent je određen na tri načina: FEM simulacija, Eulerova analitička metoda u Matlabu i blokovska shema u Simulinku. Rezultati svih triju metoda su jednaki te je moguće usporediti prednosti i mane svake metode. FEM simulacija traje nekoliko sati, dok su ostale dvije trajanja nekoliko sekundi. No, za analitičke metode potrebno je poznavanje mapa ulančenih tokova. Dodatno je napravljena analiza tranzijenta u MotorCAD-u, koja je neprecizna. Uz to, u radu je objašnjena metoda sekante za pronalazak krajnje točke demagnetizacije. To je radna točka nakon koje će se, u negativnom smjeru d-osi, magneti na rotoru demagnetizirati. Cijeli proračun je kodiran u Matlabu te je u prvom poglavlju dan pregled funkcija.

Ključne riječi: kratki spoj; sinkroni stroj s permanentnim magnetima; mape ulančenih tokova; tranzijent kratkog spoja; inverzne mape; metoda sekante; demagnetizacija

Abstract

Determination of Transient Short-Circuit Current of PM Synchronous Machines via Flux-Linkage Maps

Kristijan Gaži

The thesis analyzed transient currents and flux-linkages during a three-phase short circuit. The transient is determined in three ways: FEM simulation, Euler's analytical method in Matlab and block diagram in Simulink. The results of all three methods are equal, and it is possible to compare the advantages and disadvantages of each method. The FEM simulation lasts several hours, while the other two last for a few seconds. However, analytical methods require prepared flux-linkage maps. Transient is also determined in MotorCAD and it is concluded that it is not reliable. Additionally, the paper explains the secant method for finding the end point of demagnetization. This is the operating point after which, in the negative d-axis direction, the magnets on the rotor will demagnetize. The entire procedure is coded in Matlab, and in the first chapter an overview of the functions is given.

Keywords: short circuit; synchronous machines with permanent magnets; flux-linkage maps; short-circuit transient; inverse maps; secant method; demagnetisation

Privitak A: Programski kod

U *.zip* datoteci je priložen cjelokupni kod u Matlabu zajedno s korištenim modelima strojeva u MAGNET-u i MotorCAD-u. U nastavku je priložen ispisan kod.

Calculating direct and inverse flux-current map. Demagnetisation.

```
% Copyright 2024
%
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


delete(gcp("nocreate"))     % kill all previous multiple sessions
clear
close all
clc
```

Motor data

```
% enter motor data


motor_data.p = 2;           % number of pole pairs
motor_data.Qs = 36;         % number of slots
motor_data.a = 1;           % number of parallel branches
motor_data.In = 11.6;       % nominal current
motor_data.Un = 380;        % nominal voltage
motor_data.Lew = 0;         % total end winding inductance, H  ( zero if it is
already in model )
motor_data.Rew = 0;         % total end winding resistance, Ohm( zero if it is
already in model )
motor_data.Rs = 0.90446;    % phase resistance, Ohm
motor_data.gamma = -30;     % nominal angle gamma
motor_data.f = 50;          % frequency, Hz
```

Model data

```
model_data.poles = 1;                       % number of poles in model for
mapping
model_data.k = 8;                           % factor for scaling max current for
calculation
model_data.offset = 15;                     % offset in mechanical degrees of
axis faze A
model_data.name = 'ipm_model_Lew.mn' ;      % model name, has to be in same
folder
model_data.name_sc = 'ipm_sc_Lew_full.mn';  % model name, for short circuit
simulation
```

```
model_data.rotary = 'Rotacija';             % name of rotation object in
simulation
model_data.coilA = 'A';                     % name of coil A in model
model_data.coilB = 'B';                     % name of coil B in model
model_data.coilC = 'C';                     % name of coil C in model
model_data.period_sc = 10;                  % number of periods for short circuit
simulation
model_data.step_time_sc = 0.05;              % step time for short circuit
simulation, ms
model_data.switch_time = 10;                % switch time in ms in sc model
model_data.par_Ipeak = 'Ipeak';             % name of parameter for peak current
in sc model
model_data.par_phiA = 'phiA';               % name of parameter for current A
angle in sc model
model_data.par_phiB = 'phiB';               % name of parameter for current B
angle in sc model
model_data.par_phiC = 'phiC';               % name of parameter for current C
angle in sc model
model_data.par_f = 'f';                     % name of parameter for frequency in
sc model
model_data.par_switch = 'start';            % name of parameter for switch time
in sc model
model_data.par_Lew = 'Lew';                 % name of parameter for end winding
inductance in sc model
model_data.par_Rew = 'Rew';                 % name of parameter for end winding
resistance in sc model
model_data.Lew = 6.258e-07;                 % total end winding inductance, H  (
zero if it is already in model )
model_data.Rew = 0.4743;                    % total end winding resistance, Ohm(
zero if it is already in model )
file_name.data = 'motor_model_data.mat' ;               % file to save data
file_name.direct_map = 'IPM_direct_map' ;               % file to save direct map
file_name.direct_map3D = 'IPM_direct_map3D';            % file with saved 3D map
file_name.inverse_map = 'IPM_inverse_map';              % file to save inverse
map
file_name.sc = 'motor_SC_full' ;                        % file to save short
circuit simulation results
file_name.demag = 'IPM_demag_currents';                 % file to save max
currents during transient


n_cores = 4;                                            % number of cores for
parallelization
dimensions = [150 131];                                 % d q


save(file_name.data, 'motor_data', 'model_data', 'dimensions', 'n_cores',
'file_name');    % save variables
```

Direct map calculation

```
o = direct_flux_map(file_name.data);            % call function for direct map
calculation
if o == 1
    fprintf('Direct flux map is calculated.')
end
```

Inverse map calculation

```
inverse = inverse_flux_map(file_name.data);             % call function for
inverse map calculation
motor_inverse = inverse.grid;
inv = inverse.scat;
save(file_name.inverse_map, 'motor_inverse', 'inv');    % save inverse map to
file
fprintf('Inverse flux map is calculated.')
```

Maps plot

```
plot_map
fprintf('Maps are plotted.')
```

Short circuit simulation

```
sc = short_circuit_simulation(file_name.data);      % call function for short
circuit calculation
motor_SC = sc;
save(file_name.sc, "motor_SC");
fprintf('Short circuit simulation is done.')
```

Analytical short circuit calculation

```
motor_demag = current_calculation_demag(file_name.data);   % call function for
demagnetisation calculation
save(file_name.demag, 'motor_demag', '-append');
fprintf('Short circuit calculation is done.');
```

Transient results comparison

```
plot_transient               % plot current and flux from Euler method
fprintf('Transient is plotted.')
```

Secant method

```
motor_secant = secant(file_name.data);          % call function for secant method
% current Id needed for demagnetisation
save(file_name.demag, 'motor_secant' , '-append' );
if motor_secant.idM < motor_demag.curr(1) && -motor_secant.psiM <
motor_demag.psi(1)
    fprintf("Short circuit will not demagnetise magnets.")
else
    fprintf("Short circuit will demagnetise magnets.")
end
```

Plot secant method results

```
plot_secant
fprintf('Secant method is plotted.')
```

Direct flux map calculation

Function definition

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = direct_flux_map(fileName)
delete(gcp("nocreate"))     % kill all previous multiple sessions
```

Motor and model data

```
load( fileName )
```

Upload old file (to continue if simulation was interrupted)

```
try
    load(file_name.direct_map);             % load old file, that has only
partial solution
    start = iteration + n_cores ;           % set starting variable for
iteration, continue where it stopped
catch
    start = 0;                              % if there is no such file, start
from zero
end
```

Parallelization

```
n_par = dimensions(2);                                  % choose parallelization
and iterations dimension
if(n_cores > n_par)
        n_cores = n_par;                                % limit number of cores
if needed
        disp('Number of cores limited to the number of tasks');
end


for iteration = start : n_cores : n_par                 % start iterations for
n_cores positions
```

```
    parpool(n_cores);                                       % select number of
required cores
    par = gcp();                                            % start paralelization
```

Distribute problems

```


    if iteration + n_cores > n_par
        n_process = n_par - iteration;
    else
        n_process = n_cores;
    end
```

Function call

```
    start_time = tic;
    for i = 1 : n_process
        curr = iteration + i;
        f(i) = parfeval(par, @calc_part_new, 1, curr, dimensions, start_time,
model_data, motor_data);
    end
```

Saving results

```
    % get results from output
    par_results = cell(1,n_process);
    for idx = 1:n_process
        %// fetchNext blocks until next results are available.
        [completedIdx,value] = fetchNext(f);
        par_results{completedIdx} = value;
        % fprintf('Got result with index: %d.\n', completedIdx);
    end

    % allocate interpolants from first solution
    if iteration == 0
        motor.F_Psid=par_results{1}.F_Psid;
        motor.F_Psiq=par_results{1}.F_Psiq;
        motor.F_Psin=par_results{1}.F_Psin;
        motor.F_Tem=par_results{1}.F_Tem;
        motor.Id=par_results{1}.Id;
        motor.Iq=par_results{1}.Iq;

    end
```

```
    % append other solutions
    for i = 1 : n_process
        motor.F_Psid.Values(:,iteration + i) = par_results{i}.F_Psid.Values(:,
iteration + i);
        motor.F_Psiq.Values(:,iteration + i) = par_results{i}.F_Psiq.Values(:,
iteration + i);
        motor.F_Psin.Values(:,iteration + i) = par_results{i}.F_Psin.Values(:,
iteration + i);
        motor.F_Tem.Values(:,iteration + i) = par_results{i}.F_Tem.Values(:,
iteration + i);
    end

    delete(par);                    % stop parallel pool
    clear f                         % clear function output

    save(file_name.direct_map, 'motor', 'iteration')                % save
variables
    % show message box with simulation progress
    str = "Iterations done : " + string((iteration+n_cores)/n_cores) + " out of
"+ string(ceil(n_par/n_cores) + " . ");
    msg = msgbox(str, 'Simulation progress', 'modal');

end
out = 1;
end
```

Function definition

```
 %       Code developed for master thesis number 148, written by Kristijan
 %       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
 %       Use is permitted, with source code accordingly cited.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 function out = calc_part_new(curr, dimensions, start_time, model_data,
motor_data)

 p=motor_data.p;                                % number of pole pairs

 np_total = p*2;                                % total number of poles
 np_model = model_data.poles;                   % number of simulated poles

 a = motor_data.a;                              % number of parallel branches

 I_naz = motor_data.In;                         % nominal current
```

```
 k_Imax = model_data.k;                        % factor for scaling max current in
a map
```

```
 simulated_ratio = np_total / np_model / a;  % variable for flux multiplying,
depends on parallel branches and poles simulated
```

```
 Lew = motor_data.Lew;                         % total end winding inductance, H
 Rew = motor_data.Rew;                         % total end winding resistance, Ohm
```

Start MagNet

```
 MN = actxserver('Magnet.application');
 set (MN, 'Visible', 1);
 currentFolder = pwd;
 model = model_data.name;
 FileName = [currentFolder '\' model];
 dev = invoke(MN, 'openDocument',FileName);
 view = invoke(dev, 'getView');
 %call magnet universal constants
 Consts = invoke(MN, 'getConstants');
 %update screen automatically command
 invoke(view, 'setScaledToFit', get(Consts, 'infoTrue'));
```

Number of sample points

```
 Nd = dimensions(1);                                % d axis current
 Nq = dimensions(2);                                % q axis current

 I_peak = I_naz*sqrt(2)*k_Imax;                     % define peak value of the
phase current
```

Arrays of sample values

```
 Iq_vec = linspace(-I_peak*(7/10), I_peak*(7/10), Nq);
 Id_vec = linspace(-I_peak, I_peak/10, Nd);
```

Alocate memory

```
 motor.d.flux = zeros(Nd, Nq);
 motor.q.flux = zeros(Nd, Nq);
 motor.Tem = zeros(Nd, Nq);
```

MagNet simulations

```
 ac = exp(1i*2*pi/3);
 offset = model_data.offset;                    % offset of axis faze A
 counter = 0;
 Numpoints2 = Nd * Nq ;
```

```
 invoke(dev,'setMotionPositionAtStartup', model_data.rotary , offset)

 thetam_deg = 0;                           % angular position (mech. degrees) of
the rotating d-q reference frame relative to phase A axis
 theta_rad = thetam_deg * p * pi/180;    % angular position (electrical radians)
of the rotating d-q reference frame relative to phase A axis

     Ks_inv=[cos(theta_rad) -sin(theta_rad) 1; ...
             cos(theta_rad-2*pi/3) -sin(theta_rad-2*pi/3) 1; ...
             cos(theta_rad+2*pi/3) -sin(theta_rad+2*pi/3) 1;];       % Inverse
Park transformation matrix

         for jj = 1 : Nd
             Id = Id_vec(jj);
             mm = curr;
                 Iq = Iq_vec(mm);
                 Idq0 = [Id;Iq;0];         % dq0 current vector
                 Iabc = Ks_inv*Idq0;       % abc current vector
                 Ia = Iabc(1)/sqrt(2);
                 Ib = Iabc(2)/sqrt(2);
                 Ic = Iabc(3)/sqrt(2);

 %                 Ia = Iabc(1);
 %                 Ib = Iabc(2);
 %                 Ic = Iabc(3);

                 invoke(MN,'processCommand',['Call
getDocument().setParameter("", "CurrentOnAtTransientStart", "Yes",
infoStringParameter)']); % CurrentOnAtTransientStart is YES to turn on currents
before simulation

                 invoke(dev, 'setCoilCurrent', model_data.coilA, Ia/a, 0);
                 invoke(dev, 'setCoilCurrent', model_data.coilB, Ib/a, 0);
                 invoke(dev, 'setCoilCurrent', model_data.coilC, Ic/a, 0);
                 invoke(MN,'processCommand',['Call
getDocument().setFixedIntervalTimeSteps(0, 0, 0.1)']);
                 invoke(dev, 'solveTransient2dWithMotion');
```

```
         % Get flux linkages
```

```
invoke(MN,'processCommand',['CALL
getDocument().getSolution().getFluxLinkageThroughCoil(1,"', model_data.coilA ,'",
magnitude)']);
```

```
                invoke(MN,'processcommand',['CALL setVariant(0,magnitude,
"MATLAB")']);
```

```
                Phase_A.flux_stack = invoke(MN,'getVariant', 0, "MATLAB");
```

```
                % Total phase A flux
                Phase_A.flux = (simulated_ratio)*Phase_A.flux_stack + Lew*Ia;

                invoke(MN,'processCommand',['CALL
getDocument().getSolution().getFluxLinkageThroughCoil(1,"', model_data.coilB
,'", magnitude)']);
                invoke(MN,'processcommand',['CALL setVariant(0,magnitude,
"MATLAB")']);
                Phase_B.flux_stack = invoke(MN,'getVariant', 0, "MATLAB");

                % Total phase B flux
                Phase_B.flux = (simulated_ratio)*Phase_B.flux_stack + Lew*Ib;

                invoke(MN,'processCommand',['CALL
getDocument().getSolution().getFluxLinkageThroughCoil(1,"', model_data.coilC
,'", magnitude)']);
                invoke(MN,'processcommand',['CALL setVariant(0, magnitude,
"MATLAB")']);
                Phase_C.flux_stack=invoke(MN,'getVariant', 0, "MATLAB" );

                % Total phase C flux
                Phase_C.flux = (simulated_ratio)*Phase_C.flux_stack + Lew*Ic;

                Vector_dq_flux = 2/3*(Phase_A.flux + ac*Phase_B.flux +
ac^2*Phase_C.flux).*exp(-1i*theta_rad);

                % Calculated flux linkages d,q,f
                motor.d.flux(jj,mm) = real(Vector_dq_flux);
                motor.q.flux(jj,mm) = imag(Vector_dq_flux);
                motor.I.a(jj,mm)=Ia;
                motor.I.b(jj,mm)=Ib;
                motor.I.c(jj,mm)=Ic;
                motor.I.d(jj,mm)=Id;
                motor.I.q(jj,mm)=Iq;
```

```
                invoke(MN,'processCommand',['torque =
getDocument().getSolution().getMagneticScalarTorqueOnMotionComponent(1, "',
model_data.rotary, '")']);
                invoke(MN, 'processcommand', 'CALL setVariant(0, torque,
"MATLAB")');
                motor.Tem(jj,mm)=(np_total/np_model)*invoke(MN, 'getVariant',
0, "MATLAB");   % torque has to be multiplyed according to simulated poles
```

```
                counter = counter + 1;
```

```
                time = toc(start_time);
                rtime = round(time/counter * (Numpoints2-counter));
                [counter; Numpoints2; rtime];

        end
```

Create 3D grid variables

```
[IdX, IqY] = ndgrid(Id_vec, Iq_vec);

% Create interpolation functions
out.F_Tem = griddedInterpolant(IdX,IqY,motor.Tem,'spline');
out.F_Psid = griddedInterpolant(IdX,IqY,motor.d.flux,'spline');
out.F_Psiq = griddedInterpolant(IdX,IqY,motor.q.flux,'spline');
out.p = p;
out.Id = IdX;
out.Iq = IqY;
invoke(MN, 'processcommand','CALL close("FALSE")');
end
```

Inverse flux and current map calculation

Function definition

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function out = inverse_flux_map(fileName)


delete(gcp("nocreate"))                     % kill all previous multiple sessions
```

```
load(fileName);
load(file_name.direct_map3D);


sum.Psid = motor_skew_total.F_Psid.Values(:,:,1);
sum.Psiq = motor_skew_total.F_Psiq.Values(:,:,1);
sum.Tem = motor_skew_total.F_Tem.Values(:,:,1);
dimensions = size(motor_skew_total.F_Psid.Values(:,:,:));
for i = 2 : dimensions(3)
    sum.Psid = sum.Psid + motor_skew_total.F_Psid.Values(:,:,i);
    sum.Psiq = sum.Psiq + motor_skew_total.F_Psiq.Values(:,:,i);
    sum.Tem = sum.Tem + motor_skew_total.F_Tem.Values(:,:,i);
end


average.Psid = sum.Psid / dimensions(3) ;
average.Psiq = sum.Psiq / dimensions(3) ;
average.Tem = sum.Tem / dimensions(3) ;


data.Id = motor_skew_total.Id(:,:,1);
data.Iq = motor_skew_total.Iq(:,:,1);
data.Psi_d = average.Psid(:);
data.Psi_q = average.Psiq(:);
data.Id = data.Id(:);
data.Iq = data.Iq(:);
data.Tem  = average.Tem(:);
```

Scattered inverse maps

```
%{
get the data from maps
data.Psi_d = motor.F_Psid.Values(:,:);
data.Psi_q = motor.F_Psiq.Values(:,:);
data.Id = motor.Id(:,:);
data.Iq = motor.Iq(:,:);
data.Tem  = motor.F_Tem.Values(:,:);


% transform matrices to vectors
data.Psi_d = data.Psi_d(:);
data.Psi_q = data.Psi_q(:);
data.Id = data.Id(:);
data.Iq = data.Iq(:);
data.Tem  = data.Tem(:);
%}
% interpolate scattered inverse maps
```

```
inv.Id = scatteredInterpolant(data.Psi_d,data.Psi_q,data.Id,'natural','linear');
inv.Iq = scatteredInterpolant(data.Psi_d,data.Psi_q,data.Iq,'natural','linear');
inv.Tem  =
scatteredInterpolant(data.Psi_d,data.Psi_q,data.Tem,'natural','linear');
```

Compute the regular grid limits

```
% set boundries for gridded map
tmp   = max(motor.F_Psid.Values(:,:));
Psi_dMax = max(tmp);
tmp   = min(motor.F_Psid.Values(:,:));
Psi_dMin = min(tmp);
tmp   = max(motor.F_Psiq.Values(:,:));
Psi_qMax = max(tmp);
tmp   = min(motor.F_Psiq.Values(:,:));
Psi_qMin = min(tmp);


% create vector grids
grid.Psi_d = linspace(Psi_dMin,Psi_dMax,dimensions(1));
grid.Psi_q = linspace(Psi_qMin,Psi_qMax,dimensions(2));
[dataF.Psi_d, dataF.Psi_q] = ndgrid(grid.Psi_d, grid.Psi_q);

% fill the data
dataF.Id(:,:) = inv.Id(dataF.Psi_d(:,:),dataF.Psi_q(:,:));
dataF.Iq(:,:) = inv.Iq(dataF.Psi_d(:,:),dataF.Psi_q(:,:));
dataF.T(:,:)  = inv.Tem(dataF.Psi_d(:,:),dataF.Psi_q(:,:));
```

Map interpolation

```
motor_inverse.F_Id =
griddedInterpolant(dataF.Psi_d,dataF.Psi_q,dataF.Id,'linear','linear');
motor_inverse.F_Iq =
griddedInterpolant(dataF.Psi_d,dataF.Psi_q,dataF.Iq,'linear','linear');
motor_inverse.Psi_d =
griddedInterpolant(dataF.Psi_d,dataF.Psi_q,dataF.Psi_d,'linear','linear');
motor_inverse.Psi_q =
griddedInterpolant(dataF.Psi_d,dataF.Psi_q,dataF.Psi_q,'linear','linear');
motor_inverse.F_Tem  =
griddedInterpolant(dataF.Psi_d,dataF.Psi_q,dataF.T,'linear','linear');


out.grid = motor_inverse;
out.scat = inv ;


end
```

Short Circuit in Magnet

Function definition

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function out = short_circuit_simulation(fileName)


load(fileName)


try
    load(file_name.sc);                     % load old file, that has only
partial solution
    done = 1 ;                              % set starting variable for
iteration, continue where it stopped
catch
    done = 0;                               % if there is no such file, start
from zero
end
if ~done                                    % if not done, run simulation
```

Start Magnet and open model

```
MN = actxserver ('Magnet.application');                         % start MAGNET
set (MN, 'Visible', 1);
currentFolder = pwd;                                            % set current
folder
model = model_data.name_sc;                                     % name of the
model for short circuit
FileName = [currentFolder '\' model];                           % set full file
path
dev=invoke(MN, 'openDocument',FileName);                        % open document
via file path
view=invoke(dev, 'getView');                                    % set model view
Consts = invoke(MN, 'getConstants');                            % call magnet
universal constants
invoke(view, 'setScaledToFit', get(Consts, 'infoTrue'));        % update screen
automatically command
```

Simulation setup

```
f = motor_data.f;                                % motor frequency
offset = model_data.offset;
N_period = model_data.period_sc;
t_start = model_data.switch_time;
T = 1 / f;
omega = 2 * pi * f;


stop_time = N_period * T *1000;                  % stop time Magnet, ms
step_time = model_data.step_time_sc;             % step time Magnet, ms
t = 0 : step_time : stop_time ;
t = t/1000;                                      % time vector, s
Ipeak = motor_data.In*sqrt(2)/motor_data.a;
alpha0 = 0;
gamma = motor_data.gamma;
phi_A = 180 - alpha0 - gamma;
phi_B = 180 - alpha0 - gamma -120;
phi_C = 180 - alpha0 - gamma -240;
Rew = model_data.Rew / motor_data.a;
Lew = model_data.Lew / motor_data.a;
```

Set parameters and start simulation

```
invoke(MN,'processCommand',['Call getDocument().setFixedIntervalTimeSteps(0, ',
num2str(stop_time), ', ', num2str(step_time), ')']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_switch, '", "', num2str(t_start), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_f, '", "', num2str(f), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_Ipeak, '", "', num2str(Ipeak), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_phiA, '", "', num2str(phi_A), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_phiB, '", "',num2str(phi_B), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_phiC, '", "', num2str(phi_C), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_Lew, '", "', num2str(Lew), '", infoNumberParameter)']);
invoke(MN,'processCommand',['Call getDocument().setParameter("", "',
model_data.par_Rew, '", "', num2str(Rew), '", infoNumberParameter)']);
invoke(dev,'setMotionPositionAtStartup', model_data.rotary , offset);
invoke(dev, 'solveTransient2dWithMotion');
```

Get data

```
% get current
motor_SC.a.current = MNgetCurrent(MN, model_data.coilA) .' ;
motor_SC.b.current = MNgetCurrent(MN, model_data.coilB) .' ;
motor_SC.c.current = MNgetCurrent(MN, model_data.coilC) .' ;


% get flux linkages
motor_SC.a.flux = MNgetFluxLinkage(MN, model_data.coilA) .' ;
motor_SC.b.flux = MNgetFluxLinkage(MN, model_data.coilB) .' ;
motor_SC.c.flux = MNgetFluxLinkage(MN, model_data.coilC) .' ;


% get torque
motor_SC.Tem = getTorque(MN, model_data.rotary) .' ;
```

Calculate dq components

```
ac = exp(1i*2*pi/3);


% Flux
Vector_dq_flux = 2/3*(motor_SC.a.flux + ac*motor_SC.b.flux +
ac^2*motor_SC.c.flux).*exp(-1i*omega*t);


%Calculated flux linkages d,q
motor_SC.d.flux = real(Vector_dq_flux);
motor_SC.q.flux = imag(Vector_dq_flux);
%Current
Vector_dq_I = 2/3*(motor_SC.a.current + ac*motor_SC.b.current +
ac^2*motor_SC.c.current).*exp(-1i*omega*t);
%Calculated currents d,q
motor_SC.d.current = real(Vector_dq_I);
motor_SC.q.current = imag(Vector_dq_I);
invoke(MN, 'processcommand','CALL close("FALSE")');
out = motor_SC;
else
    out = motor_SC;         % if already calculated
end
end
```

Transient currents with Euler

Function definitions

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = current_calculation_demag(fileName)


delete(gcp("nocreate"))                 % kill all previous multiple sessions
load(fileName);                         % load motor and model data and file
names
load(file_name.direct_map);             % load inverse flux map
load(file_name.inverse_map);            % load inverse flux map
```

Motor and model data

```
Rs = motor_data.Rs;                 % motor resistance, ohm
f = motor_data.f;                    % frequency, Hz
p = motor_data.p;                   % number of poles
T = 1/f;                            % electrical period, s
n = 60*f/p;                         % mechanical speed, rpm
a = motor_data.a;                   % number of parallel branches
```

Set transient parameters

```
w = n*pi/30*p;                                      % electrical speed, rad
h = model_data.step_time_sc/1000;                   % parameter for Euler method
n_period = model_data.period_sc;                    % number of periods for
calculation
n_points = T/h;                                     % number of time points in
one period for simulation


time = linspace(0, n_period*T, n_points*n_period+1);
dt = time(2);
```

Set start values

```
iAmp = motor_data.In/a;             % nominal current
gamma = motor_data.gamma;           % nominal angle gamma
iAmp = sqrt(2) * iAmp;
iq0 = iAmp.*cosd(gamma);            % q axis current
id0 = iAmp.*sind(gamma);            % d axis current
```

```
psi_d0 = motor.F_Psid(id0, iq0);    % d axis flux
psi_q0 = motor.F_Psiq(id0, iq0);    % q axis flux
T0  = motor.F_Tem(id0, iq0);        % torque
```

Euler method for transient SC current

```
% allocate memory
i_dVect    = nan(size(time));
i_qVect    = nan(size(time));
psi_dVect  = nan(size(time));
psi_qVect  = nan(size(time));
TVect      = nan(size(time));
%set start values
i_dVect(1)    = id0;
i_qVect(1)    = iq0;
psi_dVect(1)  = psi_d0;
psi_qVect(1)  = psi_q0;
TVect(1)      = T0;
% euler equation
for tt = 2 : length(time)
    dPsi_d = (+w*psi_qVect(tt-1) - Rs*i_dVect(tt-1)) *dt;
    dPsi_q = (-w*psi_dVect(tt-1) - Rs*i_qVect(tt-1)) *dt;
    psi_dVect(tt)  = psi_dVect(tt-1) + dPsi_d;
    psi_qVect(tt)  = psi_qVect(tt-1) + dPsi_q;
    i_dVect(tt)    = inv.Id(psi_dVect(tt),psi_qVect(tt));
    i_qVect(tt)    = inv.Iq(psi_dVect(tt),psi_qVect(tt));
    TVect(tt)      = motor.F_Tem(i_dVect(tt), i_qVect(tt));
end


i_sVect = sqrt(i_dVect.^2 + i_qVect.^2);                    % total stator current
[id_transient_min, index_id_min] = min(i_dVect);            % minimal current Id,
for demag calculation
[is_transient_max, index_is_max] = max(i_sVect);            % maximal current, for
demag calculation
psi_sVect = sqrt(psi_dVect.^2 + psi_qVect.^2);              % total stator flux
[psid_transient_min, index_psid_min] = min(psi_dVect);      % minimal flux psid,
for demag calculation
[psis_transient_max, index_psis_max] = max(psi_sVect);      % maximal flux, for
demag calculation
out.curr = [id_transient_min i_qVect(index_id_min) is_transient_max ];
out.psi = [psid_transient_min psi_qVect(index_psid_min) psis_transient_max];
out.id = i_dVect;
out.iq = i_qVect;
out.psid = psi_dVect;
out.psiq = psi_qVect;
out.tem = TVect;
out.time = time;
```

Map plot

Direct maps

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load(file_name.data);
load(file_name.direct_map);
load(file_name.inverse_map)


figure()
surf(motor.Id(:,:),motor.Iq(:,:),motor.F_Psid.Values(:,:))
title('Direct \Psi_{d} map')
xlabel('I_{d}, A')
ylabel('I_{q}, A')
zlabel('\Psi_{d}, Vs')


figure()
surf(motor.Id(:,:),motor.Iq(:,:),motor.F_Psiq.Values(:,:))
title('Direct \Psi_{q} map')
xlabel('I_{d}, A')
ylabel('I_{q}, A')
zlabel('\Psi_{q}, Vs')
```

Voltage map calculation

```
Rs = motor_data.Rs;                 % motor resistance, ohm
f = motor_data.f;                    % frequency, Hz
p = motor_data.p;                   % number of poles
n = 60*f/p;                         % mechanical speed, rpm
w_m = n*pi/30;                      % mechanical speed, rad/s
w_e = w_m*p;                        % electrcal speed, rad/s
```

```
Vd_map = Rs*motor.Id(:,:) - w_e*motor.F_Psiq.Values(:,:);
Vq_map = Rs*motor.Iq(:,:) + w_e*motor.F_Psid.Values(:,:);

figure()
surf(motor.Id(:,:),motor.Iq(:,:),Vd_map(:,:))
title('Voltage V_{d} map')
xlabel('I_d, A')
ylabel('I_q, A')
zlabel('V_d, V')
```

```
figure()
surf(motor.Id(:,:),motor.Iq(:,:),Vq_map(:,:))
title('Voltage V_{q} map')
xlabel('I_d, A')
ylabel('I_q, A')
zlabel('V_q, V')
```

Torque map

```
figure()
surf(motor.Id(:,:),motor.Iq(:,:),motor.F_Tem.Values(:,:))
title('Torque map')
xlabel('I_d, A')
ylabel('I_q, A')
zlabel('T_{em}, Nm')
```

Inverse maps

```
figure()
surf(motor_inverse.Psi_d.Values(:,:),motor_inverse.Psi_q.Values(:,:),motor_invers
e.F_Id.Values(:,:))
title('Inverse I_{d} map')
xlabel('\Psi_{d}, Vs')
ylabel('\Psi_{q}, Vs')
zlabel('I_{d}, A')
figure()
surf(motor_inverse.Psi_d.Values(:,:),motor_inverse.Psi_q.Values(:,:),motor_invers
e.F_Iq.Values(:,:))
title('Inverse I_{q} map')
xlabel('\Psi_{d}, Vs')
ylabel('\Psi_{q}, Vs')
zlabel('I_{q}, A')
```

Plot transient

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load(file_name.sc)
load(file_name.demag)
load(file_name.data)


Id = motor_SC.d.current;                        % d current from FEM simulation
Iq = motor_SC.q.current;                        % q current from FEM simulation
FluxD = motor_SC.d.flux;                        % d flux from FEM simulation
```

```
FluxQ = motor_SC.q.flux;                        % q flux from FEM simulation
Tem = motor_SC.Tem;                             % torque form FEM simulation
id_transient = motor_demag.id;                  % d current from euler method
iq_transient = motor_demag.iq;                  % q current from euler method
psid_transient = motor_demag.psid;              % d flux from euler method
psiq_transient = motor_demag.psiq;              % q flux from euler method
time = motor_demag.time*1000;                   % time vector
T_el_transient = motor_demag.tem;               % torque from euler method
is_transient = sqrt(id_transient.^2 + iq_transient.^2);
psi_s = sqrt(psid_transient.^2 + psiq_transient.^2);


index_end = length(time);
moving = -model_data.switch_time/model_data.step_time_sc -1;      % to eliminate
part before short circuit
ending = index_end + moving;                    % to plot only short circuit
% move part before short circuit to the end
Id = circshift(Id, moving);
Iq = circshift(Iq,moving);
Is = sqrt(Id.^2 + Iq.^2);
FluxD = circshift(FluxD,moving);
FluxQ = circshift(FluxQ,moving);
FluxS = sqrt(FluxD.^2 + FluxQ.^2);
Tem = circshift(Tem,moving);


figure()
hold on
title('Short circuit currents transient')
plot(time(1:ending),id_transient(1:ending),'r',time(1:ending),iq_transient(1:endi
ng),'b');
plot(time(1:ending),Id(1:ending),'black', time(1:ending),Iq(1:ending),'green')
legend('I_{d,euler}','I_{q,euler}','I_{d,FEM}','I_{q,FEM}')
xlabel('t, ms')
ylabel('I, A')
grid on

figure()
plot(time(1:ending),is_transient(1:ending), time(1:ending),Is(1:ending))
title('Total current transient')
legend('I_{s,euler}','I_{s,FEM}')
xlabel('t, ms')
ylabel('I, A')
grid on

figure()
title('Short circuit flux transient')
hold on
```

```
plot(time(1:ending),psid_transient(1:ending),'r',time(1:ending),psiq_transient(1:
ending),'b');
plot(time(1:ending),FluxD(1:ending),'black',time(1:ending),FluxQ(1:ending),'green
');
legend('\psi_{d,euler}','\psi_{q,euler}','\psi_{d,FEM}','\psi_{q,FEM}')
xlabel('t, ms')
ylabel('\psi, Vs')
grid on

figure()
plot(time(1:ending),psi_s(1:ending), time(1:ending),FluxS(1:ending))
title('Total flux transient')
legend('\psi_{s,euler}','\psi_{s,FEM}')
xlabel('t, ms')
ylabel('\psi, Vs')
grid on

figure()
plot(id_transient(1:ending),iq_transient(1:ending),Id(1:ending),Iq(1:ending))
title('Short circuit currents transient')
xlabel('I_d, A')
ylabel('I_q, A')
legend('euler', 'FEM')
grid on


figure()
plot(psid_transient(1:ending),psiq_transient(1:ending),FluxD(1:ending),FluxQ(1:en
ding))
title('Short circuit flux transient')
xlabel('\psi_d, Vs')
ylabel('\psi_q, Vs')
legend('euler', 'FEM')
grid on


figure()
plot(time(1:ending),T_el_transient(1:ending), time(1:ending), Tem(1:ending))
title('Short circuit torque transient')
legend('T_{euler}','T_{FEM}')
xlabel('t, ms')
ylabel('T, Nm')
grid on
```

Secant method

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = secant(fileName)


load(fileName);
currentFolder = pwd;
myMagnetfile = model_data.name;
myMagnetpath = [ currentFolder filesep myMagnetfile];
p = motor_data.p;                                   % number of pole pairs
np_total = p*2;                                     % total number of poles
np_model = model_data.poles;                        % number of simulated poles
a = motor_data.a;                                   % number of parallel branches
simulated_ratio = np_total / np_model / a;      % parameter for flux multiplying,
depends on parallel branches and poles simulated
offset = model_data.offset;
```

Determine magnet flux and start values

```
% find magnet flux, set Id=0 and Iq=I_nominal, therefore find abs of d and q
fluxes
Id = 0;
Iq = motor_data.In/a *sqrt(2);
[Psid_M, Psiq_M] = MagnetSecant(Id,Iq, p,a, simulated_ratio, offset,
myMagnetpath, model_data);
psi_M = abs(Psid_M + 1i*Psiq_M);


% starting values of current and flux
% set two points to get two pairs (id,psid) thus line can be drawn


% FIRST point - Id=0 and Iq=0
Id = 0;
Iq = 0;
[Psid_0, ~] = MagnetSecant(Id,Iq, p,a, simulated_ratio, offset, myMagnetpath,
model_data);
psi_0 = Psid_0;
id_0 = Id;


% SECOND point - quarter nominal Id and Iq=0
Id = motor_data.In/a *sqrt(2)*sind(motor_data.gamma)/4;
Iq = 0;
```

```
[Psid_1, ~] = MagnetSecant(Id,Iq, p,a, simulated_ratio, offset, myMagnetpath,
model_data);
psi_1 = Psid_1;
id_1 = Id;
```

Iterations for secant method

```
% max number of iterations
maxIter = 10;
% allocate memory for secant method
id_secant = NaN([maxIter+2 1]);
psi_secant = NaN([maxIter+2 1]);
% set starting values in arrays
id_secant(1) = id_0;
id_secant(2) = id_1;
psi_secant(1) = psi_0;
psi_secant(2) = psi_1;
% iterations for secant method
for i = 1:maxIter
    % smooth reach to psi_M
    if i==1
        psi_max = 0;
    elseif i == 2
        psi_max = psi_M/2;
    else
        psi_max = psi_M;
    end


    id_secant(i+2) = interp1([psi_secant(i) psi_secant(i+1)], [id_secant(i)
id_secant(i+1),], -psi_max,'linear','extrap');


    [psi_secant(i+2), ~] = MagnetSecant(id_secant(i+2),0, p,a, simulated_ratio,
offset, myMagnetpath, model_data);


    % break condition, difference between calculated flux and max flux is less
than 1%
    if abs((abs(psi_secant(i+2)/psi_M)-1)*100) < 1
        break;
    end
end


% Id where max flux is reached
id_secant_demag = id_secant(i+2) * a;
```

```
% output
out.idM = id_secant_demag ;
out.id = id_secant;
out.psid = psi_secant;
out.psiM = psi_M;
end
```

Plot secant

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


load(file_name.sc);
load(file_name.demag);
load(file_name.data);


% spline secant method
id = motor_secant.id(~isnan(motor_secant.id));
psid = motor_secant.psid(~isnan(motor_secant.psid));
ii = id(1) : id(2)/5 : id(end);
pp = spline(id, psid,ii);
maxim = ceil(id(end)/-50)*-50;
iid = id(1) : id(2)/5 : maxim;
line = ones([1 length(iid)]).*(-motor_secant.psiM);
figure()
title('Secant method')
hold on
plot(motor_secant.id, motor_secant.psid,'bo')
plot(ii,pp)
plot(iid,line,'k--')
ylabel('\psi, Vs')
xlabel('I, A')
legend('','secant method','-\psi_M')
grid on


% flux plot
psid_transient = motor_demag.psid;                % d flux from euler method
psiq_transient = motor_demag.psiq;                % q flux from euler method
```

```
FluxD = motor_SC.d.flux;                                % d flux from FEM simulation
FluxQ = motor_SC.q.flux;                                % q flux from FEM simulation
moving = -model_data.switch_time/model_data.step_time_sc +1;    % to eliminate
part before short circuit
ending = length(FluxD) + moving;                    % to plot only short circuit
FluxD = circshift(FluxD,moving);
FluxQ = circshift(FluxQ,moving);


figure()
hold on
plot(psid_transient(1:ending),psiq_transient(1:ending),FluxD(1:ending),FluxQ(1:en
ding))
plot(-motor_secant.psiM,0, "Marker","square")
title('Short circuit flux transient')
xlabel('\psi_d, Vs')
ylabel('\psi_q, Vs')
legend('euler', 'FEM','psi_M')
grid on
```

MagnetSecant

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [psid, psiq] = MagnetSecant(id, iq, p,a,simulated_ratio,
offset,myMagnetpath, model_data)


hasDoc = true;
while hasDoc
   MN = actxserver('MagNet.Application');
   hasDoc = invoke(MN, 'hasDocument');
end


set (MN, 'Visible', 1);
invoke (MN, 'newDocument');
% open Document
doc=invoke (MN, 'openDocument', myMagnetpath);
invoke(doc,'setMotionPositionAtStartup', model_data.rotary, offset)
```

```
theta_deg = 0;          % angular position (mech. degrees) of the rotating d-q
reference frame relative to phase A axis
theta_rad = theta_deg * p * pi/180;      % angular position (electrical radians)
of the rotating d-q reference frame relative to phase A axis

    Ks_inv=[cos(theta_rad) -sin(theta_rad) 1; ...
            cos(theta_rad-2*pi/3) -sin(theta_rad-2*pi/3) 1; ...
            cos(theta_rad+2*pi/3) -sin(theta_rad+2*pi/3) 1;];       %Inverse Park
transformation matrix
% currrent definition
Idq0 = [id; iq; 0];
Iabc = Ks_inv * Idq0;
Ia = Iabc(1)/sqrt(2);
Ib = Iabc(2)/sqrt(2);
Ic = Iabc(3)/sqrt(2);


invoke(MN,'processCommand',['Call getDocument().setParameter("",
"CurrentOnAtTransientStart", "Yes", infoStringParameter)']); %
CurrentOnAtTransientStart is YES to turn on currents before simulation
invoke(doc, 'setCoilCurrent', model_data.coilA, Ia/a, 0);
invoke(doc, 'setCoilCurrent', model_data.coilB, Ib/a, 0);
invoke(doc, 'setCoilCurrent', model_data.coilC, Ic/a, 0);
invoke(MN,'processCommand',['Call getDocument().setFixedIntervalTimeSteps(0, 0,
0.1)']);
invoke(doc, 'solveTransient2dWithMotion');

% get flux linkages
FluxA = MNgetFluxLinkage(MN, model_data.coilA) .' ;
FluxB = MNgetFluxLinkage(MN, model_data.coilB) .' ;
FluxC = MNgetFluxLinkage(MN, model_data.coilC) .' ;


FluxA = simulated_ratio*FluxA;
FluxB = simulated_ratio*FluxB;
FluxC = simulated_ratio*FluxC;


ac=exp(1i*2*pi/3);
Flux = 2/3*(FluxA + ac.*FluxB + ac^2.*FluxC).*exp(1i.*-theta_rad);
psid = real(Flux);
psiq = imag(Flux);
invoke(MN, 'processcommand','CALL close("FALSE")');
```

```
%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of professor prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function torque=getTorque(MagNet,motionName)

invoke(MagNet,'processCommand',['position =
getDocument().getSolution().getMagneticScalarTorqueOnMotionComponent(Array(1,
Null),"',motionName,'")']);
invoke(MagNet, 'processcommand', 'CALL setVariant(0,position, "MATLAB")');
temp=invoke(MagNet, 'getVariant', 0, 'MATLAB');
if(isa(temp,'cell'))
    for i=1:numel(temp)
        torque(i)=temp{i};
    end
else
    for i=1:numel(temp)
        torque(i)=temp(i);
    end
end

%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of professor prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Current = MNgetCurrent(MagNet,coilName)

invoke(MagNet,'processCommand','ReDim magnitude(0)');
invoke(MagNet,'processCommand','ReDim phase(0)');
invoke(MagNet,'processCommand','ReDim polje(1)');
invoke(MagNet,'processCommand','polje(0)=1');
invoke(MagNet,'processCommand','polje(1)=Null');
invoke(MagNet,'processCommand',['CALL
getDocument().getSolution().getCurrentThroughCoil(polje,"',coilName,'", magnitude,
phase)']);
invoke(MagNet, 'processcommand', 'CALL setVariant(0,magnitude, "MATLAB")');
temp=invoke(MagNet, 'getVariant', 0, 'MATLAB');
if(isa(temp,'cell'))
    for i=1:numel(temp)
        Current(i,1)=temp{i};
    end
else
    for i=1:numel(temp)
        Current(i,1)=temp(i);
    end
end
%{
invoke(MagNet, 'processcommand', 'CALL setVariant(0,phase, "MATLAB")');
temp=invoke(MagNet, 'getVariant', 0, 'MATLAB');
if(isa(temp,'cell'))
    for i=1:numel(temp)
        tphase(i,1)=temp{i};
    end
```

```
else
    for i=1:numel(temp)
        tphase(i,1)=temp(i);
    end
end
%}


%       Code developed for master thesis number 148, written by Kristijan
%       Gaži, under mentorship of professor prof. dr. sc. Stjepan Stipetić.
%       Use is permitted, with source code accordingly cited.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Psi = MNgetFluxLinkage(MagNet,coilName)

invoke(MagNet,'processCommand','ReDim magnitude(0)');
invoke(MagNet,'processCommand','ReDim phase(0)');
invoke(MagNet,'processCommand','ReDim polje(1)');
invoke(MagNet,'processCommand','polje(0)=1');
invoke(MagNet,'processCommand','polje(1)=Null');
invoke(MagNet,'processCommand',['CALL
getDocument().getSolution().getFluxLinkageThroughCoil(polje,"',coilName,'",
magnitude, phase)']);
invoke(MagNet, 'processcommand', 'CALL setVariant(0,magnitude, "MATLAB")');
temp=invoke(MagNet, 'getVariant', 0, 'MATLAB');
if(isa(temp,'cell'))
    for i=1:numel(temp)
        Psi(i,1)=temp{i};
    end
else
    for i=1:numel(temp)
        Psi(i,1)=temp(i);
    end
end
%{
invoke(MagNet, 'processcommand', 'CALL setVariant(0,phase, "MATLAB")');
temp=invoke(MagNet, 'getVariant', 0, 'MATLAB');
if(isa(temp,'cell'))
    for i=1:numel(temp)
        tphase(i,1)=temp{i};
    end
else
    for i=1:numel(temp)
        tphase(i,1)=temp(i);
    end
end
```